INFO20003 Semester 2, 2025

Assignment 2: SQL

Due: Week 8 - Friday 19th September 2025, 6:00PM Melbourne Time.

Submission - Via LMS https://canvas.lms.unimelb.edu.au/

Case: The Best Movie Recommendations Ever! (BMRE!)

Inspired by the Netflix Challenge x Community Ratings x Streaming Platforms

Introduction

In the 2000s, Netflix had a contest ('The Netflix Prize')[1] to find the best ratings prediction algorithm to improve recommendations for its users. This was one of the early releases of a large real-world dataset for data scientists and machine learning/AI researchers. Since then, there are many other movie-related datasets that have been released, for both researchers, as well as for everyday movie enthusiasts (like us)!

Inspired by the Netflix Prize, we present – "The Best Movie Recommendations Ever!" 😊 – a new challenge. Assume you are part of a team of movie fans and AI experts looking to make the best recommendation algorithm[2]. Because you are the database expert in the team, you are placed in charge of the database management and querying.

IMPORTANT: You do not have to do any AI/ML stuff for this assignment, just the SQL. For the purposes of anonymity of Internet users, and respecting the intellectual property of real-world movie platforms, note that the data given for this assignment are synthetic.

[1] *Optional: See e.g., https://www.thrillist.com/entertainment/nation/the-netflix-prize*

[2] *Inspired by the exercise in F. Maxwell Harper and Joseph A. Konstan. (2015). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19.*

Description

The following description explains what data is made available to you. Be careful as many tables share identical attribute names, to simulate real-world datasets. Assume that there are no discrepancies between the different tables (e.g., Netflix year of release is the same as IMDB's year of release – see below).

Netflix records, for each movie, a unique Netflix Movie ID (e.g., 80234304), movie title, year of release. Each movie is given ratings, where a rating record consists of the numeric rating (0-5 inclusive, integers), the timestamp of the rating, and a unique anonymous user ID. For privacy, no other user data is given.

Now, in this new challenge, we also obtain data from the comprehensive Internet Movie Database (IMDB), with a record for each movie. Each IMDB movie record has an IMDB Movie ID (e.g., tt12584954), the movie title, year of release, averageIMDB rating (decimal from 0.0 to 10.0 inclusive), number of people who have rated, and classification (e.g., PG-13, or R). Each movie record is associated with a lead director and up to five main actor(s)/actress(es). Finally, each movie record has at least one genre (e.g., only Comedy, or Comedy + Drama, etc)., and one language (e.g., 'EN' for English, using the 2-letter ISO code for languages).

IMDB also uses some data from MetaCritic, which is a collection of critics scores for each movie. MetaCritic data records comprise of the IMDB Movie ID, Source, and Score (0-100 inclusive, integer); a movie can have any number of scores (e.g., one from a newspaper, the other from a film blogger).

Another reputable data source is RottenTomatoes, a movie review database. Each RottenTomatoes movie review record has a unique RottenTomatoes ID (e.g., 'the_lord_of_the_rings_the_fellowship_of_the_ring'), a 'Tomatometer' critics score (0-100 inclusive, integer), and a 'Popcornmeter' audience score (0-100 inclusive, integer).

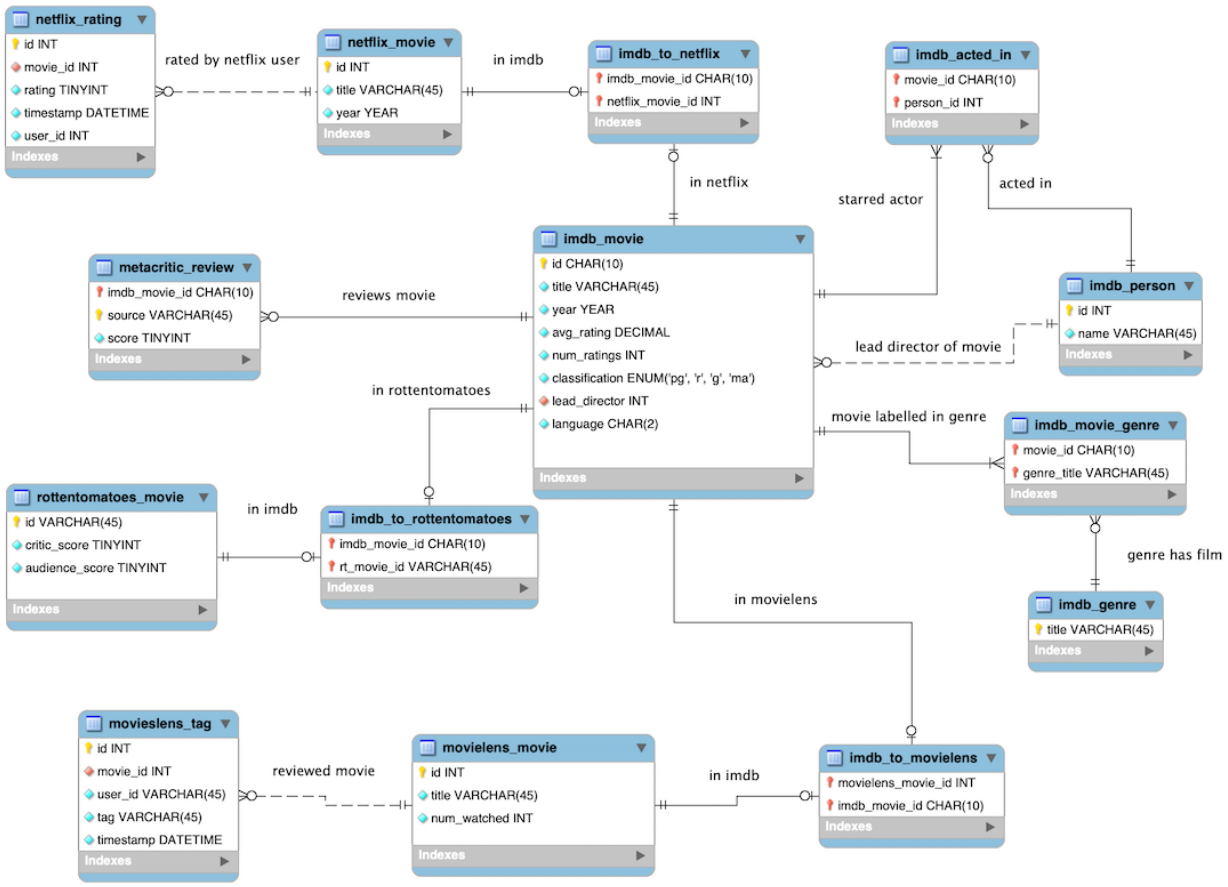
Finally, we also draw upon 'tag' data (like hashtags on social media) from the MovieLens recommendation service (inspired by Harper & Konstan, 2015). Anonymised users contribute tags to movies, in the form of records containing the MovieLens User ID, MovieLens Movie ID, the tag, and a timestamp.

Luckily, your teammates in this challenge have also supplied mappings in the following tables: imdb_to_netflix, imdb_to_rottentomatoes, and imdb_to_movielens (all self-explanatory) to link movie records across all data sources/platforms. Note that a movie *may or may not* have corresponding IDs in all tables (e.g., a new movie in cinemas may only have an IMDB record and no other corresponding records).

The Data Model

The Data Model from MySQL Workbench is provided in Figure 1.

FIGURE 1. DATA MODEL FOR BMRE!



Assignment 2 Setup

Please pay special attention to the penalties listed [⚠].

A dataset is provided which you can use when developing your solutions. To set up the dataset, download the file **BMRE.sql** from the Assignment link on Canvas and run it in Workbench. This script creates the database tables and populates them with data.

The sample dataset provided is a basic, synthetic, extract and not necessarily the 'full data'. You may find that you may need to add some more sample data in Workbench to fully test edge cases for queries.

Note that this dataset is provided for you to experiment with, but it is NOT the same dataset as what your queries will be tested against (the schema will stay the same, but the data itself may be different). This means when designing your queries, you must consider edge cases even if they are not represented in this particular data set.

The script is designed to run against your account on the Engineering IT server (info20003db.eng.unimelb.edu.au). If you want to install the schema on **your own MySQL Server installation, uncomment the lines at the beginning of the script.**

⚠ **WARNING:** ***Do NOT disable*** `only_full_group_by` ***mode when completing this assignment****. This mode is the default and is turned on in all default installs of MySQL workbench, and we've added a line to the top of slarc.sql to turn it on every time you run the script in case you disable it! You can check whether it is turned on using the command* `SELECT @@sql_mode`*;* *The command should return a string containing* `ONLY_FULL_GROUP_BY` *or* `ANSI`*.* ***When testing, our test server WILL have this mode turned on, and if your query fails due to this, you will lose marks.***

The SQL Tasks

Please pay special attention to the penalties listed [⚠].

In this section are listed 10 questions for you to answer. Write one (single) SQL statement per question. Each statement must end with a semicolon (;). Subqueries and nesting are allowed within a **single** SQL statement – however, you may be penalised for writing overly complicated SQL statements.

⚠ **WARNING: DO NOT USE VIEWS (or 'WITH' statements/common table expressions) OR VARIABLES to answer questions. Penalties apply.**

? *The Questions*

1. List all IMDB movies which contain no MetaCritic reviews. Your query should return results of the form (**IMDBMovieID, MovieTitle**). **(1 mark)**

2. Find the Netflix movie with the most recent review. Assume there are no ties: only one is the most recent. Your query should return one row (**NetflixMovieID, MovieTitle, TimeOfMostRecentRating**). **(1 mark)**

3. List all movies rated by MetaCritic source 'The Washington Post' that have at least 5 Netflix user ratings. Your query should return results of the form (**IMDBMovieID, NetflixRatingCount**). **(1 mark)**

4. Find the genre whose movies has the highest average Tomatometer score. If there are ties, then you must return all genres with the highest average. Avg score must be rounded to 1 decimal. Your query should return results of form (**genre, TomatometerAvgScore**), with one row per genre in case of a tie. **(2 marks)**

5. List all MetaCritic scores and sources for films that feature an actor with more than 2 words in their full name (e.g., "James Earl Jones", but not "Anya Taylor-Joy"). Do not duplicate results if multiple such actors acted in the same film. Your query should return results of the form (**Score, Source, IMDBTitle**). **(2 marks)**

6. Find which year has the highest number of 'pg' rated movies that have at least one MovieLens tag of 'action_thriller'. If there are ties, then you must return all results. Your query should return results of the form (**Year, MovieCount**), with one row per Year in case of a tie. **(2 marks)**

7. Find the total number of movies that are in IMDB but not in Netflix (defined as X). Similarly, find the total number of movies that are in IMDB but not in MovieLens (defined as Y). Your query should return one row: **(X, Y). (2 marks)**

8. We'll refer to a Metacritic review source that has reviewed at least one movie in *each* language that currently exists in the IMDB movies list as a 'global-reviewsource'. For each global-reviewsource, evaluate how their review count, average score, and score standard-deviation (hint: use the STDDEV operation) varies based on the language of the movie the source reviewed. Average and StdDev must be rounded to 1 decimal place. Your query should return **(globalReviewSource, language, countReviewsForLanguage, avgScoreForLanguage, popStdDevScoreForLanguage) (3 marks)**

   As an example, consider the following dataset:

   imdb_movie

id	language
1	'EN'
2	'ES'
3	'EN'

   metacritic_review

source	imdb_movie_id	score
NYT	1	3
NYT	2	5
Variety	1	2

   Currently, there are 2 unique languages in the imdb_movie table. Source 'NYT' has reviewed at least one movie in each of these languages (1x 'EN' movie and 1x 'ES' movie). Thus, NYT is a 'global-reviewsource'. 'Variety' is NOT a global-reviewsource, since they have not reviewed any 'ES' movies.

   The returned result from running our query on the given dataset should be:

Source	Language	CountReviews	AvgScore	StddevScore
NYT	EN	1	3	0
NYT	ES	1	5	0

9. A ‘multi-talented actor’ is one who has acted in movies with 5 or more genres. This includes acting in a single movie that has 5+ genres, or across multiple movies with 5+ distinct genres across all of them. Find multi-talented actors, and how many times people have tagged any of their movies. Your query should return results of the form (**ActorName, NumberOfUniqueGenres, TotalNumberOfTags**). **(3 marks)**

10. A ‘consistent quality’ movie is one where the average ratings by Netflix users (converted to a percentage from 0-100%) matches the average Metacritic rating across all sources, within 15 percentage points. For example, movie X has 3 ratings of {3,4,5} on Netflix, which converts to an average of 80.0%. Movie X has 3 MetaCritic reviews with scores {70%, 75%, 80%}, giving an average of 75.0%. Since |80.0 – 75.0| <= 15, movie X is ‘consistent quality’. If a movie doesn’t have a Metacritic review, it is not eligible to be consistent.
    Of the ‘consistent quality’ movies released in 2021 or 2022, find the 3 with the highest avg Netflix review score. If multiple movies tie, they should all be included. For example, if 4 movies all tie for 1st place, the query should return 4 rows. If 1st and 2nd were distinct, but 2 movies tied for 3rd place, the query should also return 4 rows. The final Metacritic and Netflix average scores must be rounded to 1 decimal place. Return as (**NetflixMovieID, RoundedAvgMetacriticScore, RoundedAvgNetflixScoreAsPercent**). **(3 marks)**

(This document continues, on the next page…)

⚠ *SQL Response Formatting Requirements*

Please pay special attention to the penalties listed [⚠].

To help us mark your assignment queries as quickly/accurately as possible, please ensure that:

- Your query returns the projected attributes in the same order as given in the question and does **not** include additional columns.
    - E.g., if the question asks, 'return as (userId, name)', please write `SELECT userId, name …`
    - ⚠ **DO NOT return attributes in the WRONG order,** e.g., `SELECT name, userId…`
    - You can name the columns using `` `AS` `` however you'd like, only the **order** matters. E.g., this is fine: `SELECT userId, name AS fullName`
- Please do NOT use "databaseName.tableName" format.
    - E.g., please write `"SELECT userId FROM users…"`
    - ⚠ **DO NOT provide the database name**, e.g. `SELECT userId FROM coltonc.users….`
- Ensure that you are using single quotes (') for strings (e.g. `…WHERE name = 'bob'…`) and double quotes (") only for table names (e.g. `SELECT name FROM "some table name with spaces"…`) .
    - ⚠ **Do NOT use double quotes for strings:** `…WHERE name = "bob"….`
    - ⚠ **Do NOT use Microsoft Word 'smart quotes'** (the fancy ones as you see in "this" 'example').
- Comments are optional, but we recommend writing them for complex queries.
- ⚠**Do NOT delete the special comment markers in the SQL template file.** These include: `-- BEGIN QX`, `-- END QX`, and `-- END OF ASSIGNMENT` (where X is the question number). They help us mark your submission so tampering with them will hinder our marking and **will** attract penalties.

Assignment Submission Instructions

Please pay special attention to the penalties listed [⚠].

Your submission will be in the form of an SQL script. There is a template file on the LMS, into which you will paste your solutions and fill in your student details (more information below).

This .sql file should be submitted on Canvas by **the time indicated on the first page of these instructions**.

Name your submission as 987654.sql, where 987654 corresponds to YOUR student ID number.

Please make sure that you **actually submit** your file on Canvas ⚠. After uploading the file, you need to press **'Submit Assignment'** to finalize the submission. If you submit late because you failed to press the submit button and only noticed this after the deadline, your submission will be considered late just like any other late submission to maintain fairness for all students.

Filling in the template file:

The template file on the LMS has spaces for you to fill in your student details and your answers to the questions. There is also an example prefilled script available on Canvas as well.

Below (Table 1) are screenshots from those two documents explaining the steps you need to take to submit your solutions:

TABLE 1: SCREENSHOT EXAMPLES ON HOW TO SUBMIT THE SOLUTIONS.

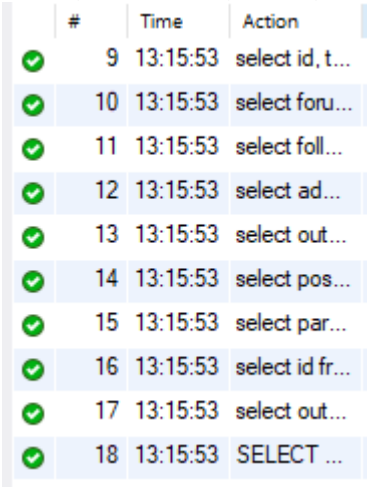
Step	***Example***

1. At the top of the template, you'll need to replace "XXXXXXXX" with your student number and name	*Template*
	`--·Your·Name:·XXXXXXXX`
	`--·Your·Student·Number:·XXXXXXX`

	Example Filled in
	`--·Your·Name:·Colton·Carner`
	`--·Your·Student·Number:·693281`

2. For each question 1-10, place your SQL solution in between the "BEGIN QX" and "END QX" markers.
Ensure each query is terminated with a semicolon ";"
	Template
	`--·BEGIN·Q1`
	`--·END·Q1`

	Example Filled in
	`--·BEGIN·Q1`
	`--·**This·is·an·example·of·a·comment·which·will·not·be·executed`
	`--·**(comments·are·not·NEEDED,·but·if·your·query·is·complex·you·can·leave·some)`
	`--·**Below·is·an·example·of·how·you·would·enter·your·answer:`
	`SELECT·*`
	`FROM·delivery`
	`NATURAL·JOIN·deliveryitem`
	`WHERE·supplierid·=·101;`
	`--·**It's·OK·to·add·more·space·in·between·the·'BEGIN·QX'·and·'END·QX'·markers`
	`--·**for·each·question,·just·don't·DELETE·the·markers!`
	`--·**Make·sure·you·fill·out·your·name·+·student·num·at·the·top·of·the·document`
	`--·**After·reading·/·understanding·these·comments·in·the·Q1·section`
	`--·**(comments·starting·with·'**'),·feel·free·to·delete·them`
	`--·**(don't·delete·lines·without·**)`
	`--·END·Q1`

3. Test that your script is valid SQL by running it from MySQL Workbench. Run the entire script by copy-pasting this entire file into a new workbench tab, placing your cursor at the start of the file (without selecting anything), and pressing the lightning bolt to run the entire file.  *All queries should run successfully one after another. If not, check to make sure you added semicolons ';' after each query.*	*All 10 queries ran sequentially and were successful.* 

Late submission

Unless you have an approved extension (see below), **you will be penalised -10% of the maximum number of marks in the assignment per calendar day that your submission is late** ⚠. For instance, if you received a 78% raw score, but submitted 2 days late, you'd receive a 58% for the assignment.

Requesting a Submission Deadline Extension

If you need an extension or special consideration due to a valid reason, you will need to comply with the Faculty of Engineering and IT (FEIT) Short Extension / Special Consideration policies and procedures.

Any requests not following the proper procedures will be rejected or returned to you. This includes requests submitted to other Faculties (e.g., Faculty of Science). Please understand that we need to always comply with FEIT policy.

Please follow the correct process and policy on Canvas -> Modules -> FEIT Extensions and Special consideration (https://canvas.lms.unimelb.edu.au/courses/215399/pages/feit-extensions-and-special-consideration?module_item_id=6885297). Then, read the following subsections for clarification.

For short extensions – please note the following:

1. If the extension has been auto-approved, you will receive an 'auto-response' email from the FEIT system. Read them carefully for the following terms:
   "Your requested extension has been granted subject to the accuracy of the information you have provided."

"Your declaration has been noted and will be sent to your subject coordinator."

2. Please keep that email as evidence/confirmation that your submission deadline extension is granted. Do not lose this email!

3. FEIT will then inform us, and we will *manually* update your due date on Canvas.

4. Please note that the FEIT system does *not* connect to Canvas, hence each change must be done manually. We appreciate your patience as we often have many cases to go through.

For Special Consideration – please note the following:

We quote the following advice from FEIT:

"Extension requests between 4 and 10 calendar days must have appropriate supporting documentation and be submitted via the Special consideration portal. ... Students with AAPs also need to apply for special consideration through Special consideration portal with supporting documents"

These will be processed centrally on a case-by-case basis, and you will receive a formal outcome via email.

Note: Short Extension Requests received after the assignment deadline will be rejected, based on the requirements of the Special Consideration policy (https://eng.unimelb.edu.au/students/coursework/study-resources/extensions-and-special-consideration):
If students need to request an extension longer than 4 days or submit the request after the assessment submission deadline, the application must have appropriate supporting documentation and be submitted via the Special consideration portal.

(In other words, any extensions which fall within the scope/criteria of the Special Consideration policy needs to be assessed formally and centrally by the University administration and not by the teaching team.)

Reminder: INFO20003 Hurdle Requirements

To pass INFO20003, you must pass two hurdles:

- **Hurdle 1:** Obtain at least 50% (15/30) or higher for the three assignments (each worth 10%)
- **Hurdle 2:** Obtain at least 50% (35/70) or higher for the combination of quizzes and end of semester exam
  It is our recommendation to students that you attempt every assignment and every question in the exam.

Reminder: Academic Integrity

"You are not permitted to use an AI tool in any way when completing your work. Any use suspected will be reported as potential academic misconduct and subject to appropriate penalties."

⚠ **Students are kindly reminded to always comply with the University's Academic Integrity Policies.**

Please read https://academicintegrity.unimelb.edu.au/plagiarism-and-collusion and please note:
"If a student is found to have deliberately plagiarised or colluded the penalties are severe and can include failure of a subject or exclusion from the University. The University provides extensive resources and educates students about academic integrity so that students are aware of what constitutes plagiarism and collusion, and the consequences of those practices."
(Source: https://academicintegrity.unimelb.edu.au/plagiarism-investigation-and-penalties)

⚠ **Note that you will have SQL questions in the exam as well, which are probably tougher than these ones here: cheating will not help you in the exam. So please, no LLM or other AI assistance within this assignment.**

⚠ **We will also be using a code similarity checker to compare student submissions against other submissions, as well as against LLM-generated code.**

GOOD LUCK!