1. Motivation

因為我喜歡看電影，但是常常又不知道要怎麼選擇該看哪一部電影，而且一部一部的從 IMDB(國外的電影推薦網站)去人工搜索又很浪費時間，因此就決定來做一個從 database 裡面找到符合使用者想看的電影的 application.

2. Application Description

程式開啟後會有一段前言告訴你這是一個 daily recommendation of movie 的程式，並給你一個範例的 constraint，可以利用 rating, language, runningtime(電影長度)來篩選 database 裡面的資料，然後用 javascript 處理好從 database 接收的所有條目(json 格式)，轉成 string 後儲存成 array，再用 random 來隨機推薦一部電影，並列出這部電影所有相關的訊息，如果 constraint 有不符合格式則會跳出”incorrect query!” 來要求使用者重新輸入，或是使用者輸入的 constraint 太嚴格，以至於 database 中沒有符合的電影推薦給使用者，則會跳出“sorry, there’s no movie for your preference today:(” 要求使用者把 constraint 修改一下.

3. Data sources and how to collect and import data

Here is the data source from. As there's some movie title written in non-English characters, it is decoded wrong as UTF-8 and cannot be displayed correctly, so I delete those movies by preprocessing in the csv file and then use "import data from csv file" in the postgresql table. Can simply create another csv file and then import to update the database.

4. Database Schema

recommendation
Title(PK)
Genre
Premiere
Runningtime
Rating
language

Which is already a BCNF table. Indexing on rating, as I expect almost every user will use rating as constraint.

5. The application’s functions and related queries

The application requires the users to input constraint on their own to translate it into a real SQL query, the query eventually translated by the javascript program and further sent into the database on the AWS server is in the form of “select * from “recommendation” where [user input constraint].” If user input “language” constraint, it will be processed in the javascript by separating it into “(language like [lang] or language like [lang])” (see further in the attached javascript code on E3).

6. Demo Video

Attached on E3.

7. To open the application, you can use cmd prompt and type “node index.js” to execute the javascript file.