Defining and Using Views Exercises

You will define (virtual) views over the movie-ratings database that was also used for the SQL Movie-Rating Query Exercises, and you will write queries that reference the views instead of or in addition to the base tables. A SQL file to set up the schema and data for the movie-ratings database is downloadable here. This schema and data can be loaded as specified in the file into SQLite, MySQL, or PostgreSQL; see our quick guide for installing and using these systems. These exercises can be performed on any of the three systems.

Schema:
Movie (mID, title, year, director)
English: There is a movie with ID number *mID*, a *title*, a release *year*, and a *director*.

Reviewer (rID, name)
English: The reviewer with ID number *rID* has a certain *name*.

Rating (rID, mID, stars, ratingDate)
English: The reviewer *rID* gave the movie *mID* a number of *stars* rating (1-5) on a certain *ratingDate*.

Each exercise asks you to create a view, and then write a query using that view, perhaps along with previously created views and/or the base tables. The correct results for the queries over the provided data can be seen by pressing the button at the bottom of the page.

1. Create a view called *TNS* containing title-name-stars triples, where the movie (title) was reviewed by a reviewer (name) and received the rating (stars). Then referencing only view *TNS* and table *Movie*, write a SQL query that returns the lastest year of any movie reviewed by Chris Jackson. You may assume movie names are unique.

2. Referencing view *TNS* from Exercise 1 and no other tables, create a view *RatingStats* containing each movie title that has at least one rating, the number of ratings it received, and its average rating. Then referencing view *RatingStats* and no other tables, write a SQL query to find the title of the highest-average-rating movie with at least three ratings.

3. Create a view *Favorites* containing rID-mID pairs, where the reviewer with rID gave the movie with mID the highest rating he or she gave any movie. Then referencing only view *Favorites* and

tables *Movie* and *Reviewer*, write a SQL query to return reviewer-reviewer-movie triples where the two (different) reviewers have the movie as their favorite. Return each pair once, i.e., don't return a pair and its inverse.

Hide Query Results

1. 1982

2. Raiders of the Lost Ark

3. *These tuples can be returned in any order, and it's okay if the reviewer names are reversed*
(Sarah Martinez, Mike Anderson, Gone with the Wind)
(Daniel Lewis, Elizabeth Thomas, Snow White)
(Brittany Harris, Chris Jackson, Raiders of the Lost Ark)