Carnegie Mellon University

15-415 Database Applications

Spring 2012, Faloutsos

Assignment 2: SQL

Due: 2/14, 1:30 pm, in class – hard copy *and* electronic files

Reminders

- Weight: **15%** of the homework grade.
- The points of this homework add up to **100**.
- Like all homeworks, this has to be done individually.
- Lead TA: Bin Fu.
- Rough time-estimates: **4~6 hours**.

What to hand in

1. A **hard-copy** of the SQL statements (including any views defined) and all the query outputs. Submit in-class.
2. A SQL file for each question that generates the desired output - so you will have 10 SQL query files named query1.sql, query2.sql and so on. Email a **zip archive** your-andrew-id.zip containing all the SQL query files to binf@andrew.cmu.edu with the subject [15-415 HW2 your-andrew-id]. We plan to run your scripts and diff the output against correct answers.

Notes

1. Feel free to use views (but of course give the SQL definitions before using them). But please avoid creating additional tables.
2. There may be more than one correct answer; we will accept them all.

Database Setup

We will work with PostgreSQL - a popular open-source database. We have put up a readme file on the course homepage with detailed instructions; link:
http://www.cs.cmu.edu/~christos/courses/dbms.S12/hws/HW2/PostgreSQLReadme.htm

Here's the quick summary to get you started:

1. Log into `newcastle.db.cs.cmu.edu` using your-andrew-id+415 as your user-id, e.g., for andrew-id binf, the user-id is binf415. The initial password of your account is same as the user-id e.g., for andrew-id binf, password is binf415. You will be prompted to change the password immediately after the first login, by *first entering the initial password, and then giving your new password twice*. Please pick a strong new password which meets the password safety requirement of the server machine (like, YW9c10bK etc.).
2. On first login, run `../binf415/setup_db.sh`, press "y" to continue when prompted.
3. Run `pg_ctl start -o -i`, then press "Enter".
4. Run `psql`.
5. Run `SELECT COUNT(*) FROM movies`, the count should equal 2,680**.**
6. Run `SELECT COUNT(*) FROM play_in`, the count should equal 74,772.
7. Run `\q` to quit PostgreSQL.
8. IMPORTANT: Please stop the server using `pg_ctl stop` before logging out.

Question: SQL queries on the MovieLens dataset (100 points)

Problem Description

In this question we will use the MovieLens dataset released in May 2011. For more details, please refer to:
http://www.cs.cmu.edu/~christos/courses/dbms.S12/hws/HW2/movielens-readme.txt
We preprocessed the original dataset and loaded the following two tables to PostgreSQL:

> **movies** (**mid**, **title**, **year**, **num_ratings**, **rating**)
> **play_in** (**mid, name**, **cast_position**)

In the table **movies**, **mid** is the unique identifier for each movie, **title** is the movie's title, and **year** is the movie's year-of-release. Each movie receives a total number of **num_ratings** ratings from users, and the average rating is **rating** on a scale of 0.0-10.0.

The table **play_in** contains the main cast of movies. **name** is the actor's name (assume each actor has an unique name). **cast_position** is the order of the actor where he/she appears on the movie cast list (For example, in the movie *Titanic*, the **cast_position**s of *Leonardo DiCaprio* and *Kate Winslet* are 1 and 2, respectively).

Queries

Write SQL queries for the following:

[**Q1 – Actor Pool**] Find the total number of distinct actors in the database. [**5 points**]

[**Q2 – Best Rated Motion Pictures**] Among all the movies rated for at least 10,000 times, select the top-10 movies (**mid**, **title**, and **rating**) which receive the highest average rating. [**5 points**]

[**Q3 – Largest Crew**] Return the movie (**mid**, **title**) that employed the largest number of actors. If there is a tie, sort the **mid**'s in ascending order. [**10 points**]

[**Q4 – Active Actor Pool**] Return the total number of *active actors* in the database. An *active actor* is one whose most recent movie is released after the year 2006 (i.e. **year**>2006). [**10 points**]

[**Q5 – Working with *Cruise***] Return the actors (**name**) who has collaborated with *Tom Cruise* the most of times. If there is a tie, sort the **name** in ascending order. [**10 points**]

[**Q6 – Working with the New Couple**] Return the actors (**name**) who have collaborated with **both** *Tom Cruise* and *Katie Holmes*. Sort the output on ascending order of **name**. [**10 points**]

[**Q7 – The Most Famous Actors**] Find the top-10 *active actors* (as defined in Q4) whose movies receive the most ratings in total. For example, if *Tom Hanks* had played in three movies, whose **num_rating**s are 30, 20, and 10 respectively, then the total number of ratings of his movies would be 30+20+10=60. [**10 points**]

[**Q8 – The Best Actors in a Leading Role**] We define the *leading role* of each movie as the actors whose **cast_position** is either 1 or 2 (so the leading role of the movie *Titanic* is *Leonardo DiCaprio* and *Kate Winslet*). Then we say the *star quality* of each actor is the average rating of the movies in which he/she has played a leading role. For example, if *Leonardo DiCaprio* had acted in three movies, whose **rating**s are 7.0, 7.5 and 8.0 respectively, but he only played the first two movies as a leading role, then his *star quality* would be (7.0 + 7.5) / 2 = 7.25.

Find the actors (**name**) with top-10 star quality. To reduce noise, the actor must have played as a leading role for at least 5 times. [**10 points**]

[**Q9 – The Breakout Actors in 2008**] Find the number of actors who have acted in more movies in the year 2008 than any other year in their career. [**15 points**]

[**Q10 – Dual Favorite Movies**] One analyst wants to find a set of all-time best movies. He considers two metrics at the same time: its average rating (**rating**) and the number of ratings it receives (**num_ratings**). He uses the following rule to determine whether one movie is *better* than another: For two movies m_1 and m_2, we define that m_1 ***dominates*** m_2 if and only if m_1 has a higher average rating *and* m_1 receives more ratings.

Find a set of movies (**title**, **num_ratings, rating**) M, so that each movie in M is not dominated by any other movie in the database. Return the output on ascending order of **title**. (FYI: This query is also known as the *Skyline Query*) [**15 points**]

Reminder: To quit PostgreSQL, use `\q`. Please stop the server using `pg_ctl stop` before logging out after you complete the assignment.