Carnegie Mellon University
15-415 - Database Applications
Fall 2009, Faloutsos
Assignment 3: SQL

Due: 9/22, 1:30pm

Reminders:

- Weight: **10%** of the homework grade.
- Out of **100** points.
- Lead TA: B. Aditya Prakash
- Estimated time to complete the assignment is **6 hours**.

What to hand in:

1. A hard-copy of the SQL statements (including any views defined) and all the query outputs. Also include a hard-copy of the plot for Q1.5. Submit in-class.
2. A SQL file for each query of Question 1 that generates the desired output - so you will have 11 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 badityap AT cs.cmu.edu with the subject [15-415 HW3 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. Please post your questions to the bboard (`http://www.cmu.edu/blackboard/`) or send them to badityap AT cs.cmu.edu.
3. 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-F09/hws/hw3/PostgreSQL_Readme.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 `badityap`, the user-id is `badityap415`. The initial password of your account is same as the user-id *e.g.*, for andrew-id `badityap`, password is `badityap415`. 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 ∼`badityap415/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 97.
6. Run `SELECT COUNT(*) FROM ratings`, the count should equal 35,000.
7. Run `\q` to quit PostgreSQL.
8. **IMPORTANT:** Please stop the server using `pg_ctl stop` before logging out.

Question 1. Movies and Ratings Redux (100 points)

Problem Description

In this question we will (again!) work with a subset of the Netflix database which contains the following 2 tables:

```
movies (mid, title, year)

ratings (mid, cid, rating, timestamp)
```

where `mid` is the unique id for each movie, `title` is the movie's title, `year` is the movie's year-of-release and `cid` is the unique id for each customer. The attribute `rating` is an integer ranging between 1 and 5 (and as in Homework 1, a rating of 5 means the customer loved the movie and a rating of 1 implies he hated it). The `timestamp` denotes the time when the customer rated the movie.

Definitions

Let's first define a few terms which will be helpful later:

- *Activity level* of a customer is the number of movies he/she rated.
- *Popularity* of a movie is the number of ratings the movie received.
- A *top-rated* movie refers to the movie which has the maximum average rating.

Queries

Write SQL queries for the following:

[**Q1.1**] Find the most active customer (`cid`) and his activity level in the database. [**5 points**]

[**Q1.2**] Find the total number of distinct customers. [**5 points**]

[**Q1.3**] Find the top-10 rated movies (`mid` and `title`; sorted ascending order on `mid`). [**5 points**]

[**Q1.4**] For each movie give its `mid` and average rating. Movies which have not been rated should also be present in the output (with a `null` for the average rating). Output should be sorted in ascending order on `mid`. [**5 points**]

[**Q1.5 - Ratings Histogram**] For each specific rating ({1-5}), give the count of movies in the database with that rating. The output should be sorted ascending order on rating. Also plot the values with the ratings on the x-axis and count of movies on the y-axis and submit a hardcopy of the plot. *For no points: before computing the numbers, what would you expect the plot to look like (Gaussian?)? And after plotting it, can you explain the curve?* [**10 points**]

[**Q1.6 - March Madness**] We are interested in movies that got at least 100 ratings during March 2009. For each such movie give its `mid` and the average rating it got in March 2009. Again the `mid`'s should be in sorted ascending order in the output. [**10 points**]

[**Q1.7 - Peers**] Find all the *peers* of the customer with `cid` = 19002 (we'll call him 'Bob'). A peer for 'Bob' is a customer who has rated at least one movie with exactly the same rating as 'Bob'. The list of peers should be in sorted ascending order on the `cid`. [**10 points**]

[**Q1.8 - Recommendation**] We want to recommend a new movie for 'Bob'. Excluding the movies he has rated, we want to find the movie that his peers like the most i.e. write a query to find the `title`, `mid` and `year` for the movie with the highest average rating among his peers. [**10 points**]

[**Q1.9 - Outliers**] We want to find people who gave bad scores to the highest rated movies. Specifically, find the user whose average rating for the top-5 rated movies was the least. If there is a tie, give the `cid`'s in ascending order. [**10 points**]

[**Q1.10 - Millennium-Movie Lovers**] Find the customers (`cid`, in ascending order) who like movies released in the year 2000 the most. That is, if we compute their average rating for movies per year-of-release, then the year-of-release 2000 got the highest average rating from each of them (or the year 2000 was tied in first place). [**15 points**]

[**Q1.11 - Possible Bot**] Find the user with the grand maximum number of ratings per day i.e. compute the activity level per day of each user and find the user with the maximum. Give the user `cid` as well as the corresponding maximum. If there is a tie, print all in ascending `cid` order. [**15 points**]

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