Carnegie Mellon University
Department of Computer Science
15-415/615 - Database Applications
C. Faloutsos & A. Pavlo, Spring 2014

Homework 2

IMPORTANT - what to hand in:

- Please submit your answers in
  - **hard copy** in class at **1:30pm on Tuesday, February 11th**. For each question, we want both your SQL expression, *as well as* the output of your query on the provided database.
  - And **electronically** on the "blackboard", by the same date/time.

  The electronic submission should be an archive called `[andrew_id].zip`, containing a folder called `queries` and a file for each query, called `q#.sql`, where '#' is the query number. For your convenience, mirror the structure of the file `http://www.cs.cmu.edu/~epapalex/15415S14/db_hw2_s14.zip`, where you can replace the place-holder scripts `/queries/q#.sql` with your real answers.

Reminders:

- **Plagiarism:** Homework may be discussed with other students, but all homework is to be completed **individually**.
- **Typeset** all of your answers whenever possible. Illegible handwriting or ambiguous answers may get no points, at the discretion of the graders.
- For faster grading, please solve each of the 8 questions on a **separate** page, i.e., 8 pages in total for this homework. Type **course-id, hw-id, question-id**, and your **name and Andrew ID on each** of the 8 pages.
- **Late Homeworks**: If you are turning your homework in late, please email it
  - to all TAs
  - with the subject line exactly `15-415 Homework Submission (HW 2)`
  - the count of slip-days you are using, and the count you have left.

For your information:

- Graded out of **100** points;
- **8** questions total

- Rough time estimate: 8 hours (1 hour for each question on average - some questions require less time, some more. Points assigned to each question correlate well with the time needed to solve it.)

Revision : 2014/02/21 23:55

Question	Points	Score
Warm-up queries	5	
Find the star's movies	5	
Popular actors	15	
Most controversial actor	10	
The minions	20	
High productivity	5	
Movies with similar cast	15	
Skyline query	25	
Total:	100	

Introduction

In this homework you will have to write SQL queries to answer questions on a movie dataset, about movies and actors. The database contains two tables:

- `movies`(mid, title, year, num_ratings, rating)
- `play_in`(mid, name, cast_position)

The tables contain the obvious information: which actor played in what movie, at what position; for each movie, we have the title (eg., 'Gone with the wind'), year of production, count of rating reviews it received, and the average score of those ratings (a float in the range 0 to 10, with '10' meaning 'excellent').

We will use Postgres, which is installed in the andrew machines. [1]

You need to do the following set up steps.

1. **Get recommended machine and port number:** On the "blackboard" grades table there are two columns, called `ghc##` and `PGPORT`. The first is the number of the `ghc` machine we recommend for you, and the other is the recommended port number for your Postgres server instance. The goal is to avoid conflicts, by having each of you running on a different machine, and listening on a different port. To log in to the recommended machine, type

   ```
   ssh ghc##.ghc.andrew.cmu.edu
   ```

   replacing `##` by the two digit number you found on "blackboard".

   In the remote case of conflicts, choose a port number close to the one assigned to you and try again. The GHC machines from 02 to 81 are located in GHC 3000, 5201, and 5205 and can be accessed through `ssh`. There are also machines in GHC 5208 which can be accessed physically.

2. **Get and edit setup script**: Download the HW2 .zip from here:

   `http://www.cs.cmu.edu/~epapalex/15415S14/db_hw2_s14.zip`.

   Unzip it and copy the folder `db_hw2_s14` (make sure the name is exactly this one) it on your *home directory* of your andrew account. Edit `setup_db.sh` to modify the line that assigns a value to `PGPORT` to the value that you found on blackboard.

3. **Set up the db:** If your default shell is not bash, type `bash` and then

   ```
   bash setup_db.sh.
   ```

   This will start the Postgres server, create a database with your username, create the schema for this homework, and add data to the database.

4. **Test script:** Run

   ```
   bash run_queries.sh.
   ```

[1] You may develop your queries on your own, local installation of Postgres, but your solutions will be run and graded on the public installation.

This shell script executes all the queries for this homework (which you have to implement). Inside the folder you unzipped, there is a folder called `queries` which contains place-holder .sql files for all the queries that you have to implement. Replace the place-holders with *your* queries, and submit them in a zip file, as we said in the first page.

5. **Optional testing:** If you want to test/debug queries separately, you can also use the script `test.sh`, included in the homework material.

6. **Starting/stopping** the Postgres server: see the instructions at

   `http://www.cs.cmu.edu/~epapalex/15415S14/PostgreSQLReadme.htm`.

The full documentation for Postgres is at http://www.postgresql.org/docs/.

Question 1: Warm-up queries . . . . . [5 points]

Submit on separate page

(a) **[2 points]** Print all actors in the movie `Quantum of Solace`, sorted by cast position. Print only their names.

(b) **[3 points]** Print all movie titles that were released in 2002, with `rating` larger than 8 and with more than one rating (`num_ratings` > 1).

Solution:

(a)
```
SELECT name FROM play_in p, movies m
WHERE p.mid = m.mid and m.title='Quantum of Solace'
ORDER BY p.cast_position;

             name
-----------------------------
 Daniel Craig
 Olga Kurylenko
 Mathieu Amalric
 Judi Dench
 Giancarlo Giannini
 Gemma Arterton
 Jeffrey Wright
 David Harbour
 Jesper Christensen
 Anatole Taubman
 Rory Kinnear
 Tim Pigott-Smith
 Fernando Guillen-Cuervo
 Jesus Ochoa
 Glenn Foster
 Paul Ritter
 Simon Kassianides
 Stana Katic
 Lucrezia Lante della Rove...
 Neil Jackson
 Oona Chaplin
(21 rows)
```

(b)
```
SELECT title FROM
movies
WHERE year = 2002 and rating>8 and num_ratings>1;

                  title
```

```
-------------------------------------
 The Lord of the Rings: The Two Towers
 Cidade de Deus
 Mou gaan dou
(3 rows)
```

Grading info: *Graded by Alex Beutel*

- *-1 for not ordering the actors correctly*
- *-1 for each question in which the command is given but no result*
- *Note: there is no need to do a join of any sort (or even touch the play_in table at all) in part b.*
- *-0.5 if print whole row not just titles for part b*

Question 2: Find the star's movies . . . . . [5 points]

Submit on separate page

(a) **[5 points]** Print movie titles where `Sean Connery` was the star (i.e. he had position 1 in the cast). Sort the movie titles alphabetically.

Solution:

```
SELECT title from movies m, play_in p
WHERE m.mid = p.mid and name = 'Sean Connery' and cast_position = 1
ORDER BY title;

                 title
--------------------------------------
 Der Name der Rose
 Diamonds Are Forever
 Dr. No
 Entrapment
 Finding Forrester
 First Knight
 From Russia with Love
 Goldfinger
 Never Say Never Again
 The Hunt for Red October
 The League of Extraordinary Gentlemen
 Thunderball
 You Only Live Twice
(13 rows)
```

Graded by Alex Beutel

Grading info:

- *-1 for not ordering the titles correctly*
- *-1 if the command is given but no result*

Question 3: Popular actors ..... [15 points]

Submit on separate page

(a) **[8 points]** We want to find the actors of the highest quality. We define their quality as the weighted average of the ratings of the movies they have played in (regardless of cast position), using the number of ratings for each movie as the weight. In other words, we define the quality for a particular actor as

$$\frac{\sum_{all\ movies\ of\ actor}(\texttt{num\_ratings} \times \texttt{rating})}{\sum_{all\ movies\ of\ actor} \texttt{num\_ratings}}$$

Print the names of the top 5 actors, according to the above metric. Break ties alphabetically.

(b) **[7 points]** Now we want to find the 5 most popular actors, in terms of number of ratings (regardless of positive or negative popularity). I.e, if actor 'Smith' played in 2 movies, with `num_ratings` 10 and 15, then Smith's popularity is 25 (=10+15). Print the top 5 actor names according to popularity. Again, break ties alphabetically.

Solution:

(a)
```
DROP VIEW IF EXISTS WeigthedRatings;

CREATE VIEW WeightedRatings AS
SELECT name, SUM(rating*num_ratings)/SUM(num_ratings) AS WeightedRating
FROM movies m, play_in p WHERE m.mid = p.mid GROUP BY(name);

SELECT name FROM WeightedRatings
ORDER BY
WeightedRating DESC,
name ASC
LIMIT 5;

         name
-----------------------
 Adam Kalesperis
 Aidan Feore
 Aleksandr Kajdanovsky
 Alexander Kaidanovsky
 Alisa Frejndlikh
(5 rows)
```

(b)
```
DROP VIEW IF EXISTS ActorSumRatings;
```

```
CREATE VIEW ActorSumRatings AS
SELECT name, SUM(num_ratings)  as popularity
FROM play_in p, movies m
WHERE p.mid = m.mid
GROUP BY name;

SELECT name from ActorSumRatings
ORDER BY popularity DESC, name ASC
LIMIT 5;

         name
----------------------
 Johnny Depp
 Alan Rickman
 Orlando Bloom
 Helena Bonham Carter
 Matt Damon
(5 rows)
```

Grading info: *Graded by Vagelis Papalexakis*

- *-1 for not ordering the titles correctly*
- *-1 if the command is given but no result*
- *-1 if the formula for popularity is wrong*
- *half points if no hard copy and/or makefile included*

Question 4: Most controversial actor ..... [10 points]

Submit on separate page

(a) **[10 points]** We want to find the most controversial actor. As a measure of controversy, we define the maximum difference between the ratings of two movies that an actor has played in (regardless of cast position). That is, if actor 'Smith' played in a movie that got `rating`=1.2, and another that got `rating`=9.5, and all the other movies he played in, obtained scores within that range, then Smith's contoversy score is 9.5-1.2= 8.3. Print the name of the top-most controversial actor - again, if there is a tie in first place, break it alphabetically.

Solution:

```
DROP VIEW IF EXISTS RatingGap;

CREATE VIEW RatingGap AS
SELECT p1.name, MAX(ABS(m1.rating-m2.rating)) as Gap
FROM play_in p1, play_in p2, movies m1, movies m2
WHERE p1.mid = m1.mid and p2.mid = m2.mid  and p1.name = p2.name
GROUP BY(p1.name);

SELECT name FROM RatingGap ORDER BY(Gap) DESC LIMIT 1;

     name
---------------
 John Travolta
(1 row)
```

Grading info: *Graded by Vagelis Papalexakis*

- *-1 for not ordering the titles correctly*
- *-1 if the command is given but no result*
- *half points if no hard copy and/or makefile included*

Question 5: The minions ..... [20 points]

Submit on separate page

(a) **[20 points]** Find the "minions" of `Annette Nicole`: Print the names of actors who only played in movies with her and never without her. The answer *should not* contain the name of `Annette Nicole`. Order the names alphabetically.

Solution:

```
DROP VIEW IF EXISTS MastersMovies CASCADE;

CREATE VIEW MastersMovies AS
SELECT m.mid,m.title  FROM movies m, play_in p
WHERE m.mid = p.mid and p.name = 'Annette Nicole';

DROP VIEW IF EXISTS CoActors;

CREATE VIEW CoActors AS
SELECT DISTINCT name FROM MastersMovies m , play_in p
WHERE p.mid = m.mid;

DROP VIEW IF EXISTS Combinations;

CREATE VIEW Combinations AS
SELECT name,mid FROM MastersMovies , CoActors;

DROP VIEW IF EXISTS NonExistent;

CREATE VIEW NonExistent AS
SELECT * FROM Combinations
EXCEPT
(SELECT name, mid FROM play_in);

DROP VIEW IF EXISTS PotentialResults;

CREATE VIEW PotentialResults AS
SELECT * from CoActors
EXCEPT
(SELECT distinct(name) FROM NonExistent);

DROP VIEW IF EXISTS NotMastersMovies;

CREATE VIEW NotMastersMovies AS
SELECT m.mid FROM movies m
```

```
EXCEPT
(SELECT mid FROM MastersMovies);

SELECT * from PotentialResults
WHERE name not in
(SELECT name
FROM play_in p, NotMastersMovies m
WHERE m.mid = p.mid
UNION SELECT 'Annette Nicole'
)
ORDER BY name;

      name
-----------------
 Christian Perry
(1 row)
```

Grading info: *Graded by Wang Shen*

- *Most of the students understand this question in a different way (find all the actors who only play with Annette but not necessarily play in all of Annette's movies). Because of the ambiguity of this problem, both solutions will be accepted*
- *-2 for not following the requirements of the question (e.g. wrong order, including Annette in result, etc.)*
- *-2 for not removing duplicated results*
- *If the result is mostly incorrect, in general half of the points will be taken*

Question 6: High productivity . . . . . [5 points]

Submit on separate page

(a) **[5 points]** Find the top 2 most productive years (by number of movies produced). Solve ties by preferring chronologically older years, and print only the years.

Solution:

```
DROP VIEW IF EXISTS MoviesPerYear;

CREATE VIEW MoviesPerYear AS
SELECT year, COUNT(title) num_movies
FROM MOVIES GROUP BY(year);

SELECT year from MoviesPerYear ORDER BY num_movies DESC LIMIT 2;

 year
------
 2006
 2007
(2 rows)
```

Grading info: *Graded by Wang Shen*

- *-1 for not following the requirements of the question (e.g. wrong order, wrong limit, etc.)*
- *If the result is mostly incorrect, in general half of the points will be taken*

Question 7: Movies with similar cast ..... [15 points]

Submit on separate page

(a) **[8 points]** Print the count of distinct pairs of movies that have at least one actor in common (ignoring cast position). Exclude self-pairs, and mirror-pairs.

(b) **[7 points]** Print the count of distinct pairs of moves that have at least two actors in common (again, ignoring cast position). Again, exclude self-pairs, and mirror-pairs.

Solution:

(a)
```
SELECT COUNT(*) FROM (SELECT DISTINCT m1.mid, m2.mid
FROM movies m1, movies m2, play_in p1, play_in p2
WHERE m1.mid > m2.mid and m1.mid = p1.mid and
m2.mid = p2.mid and p1.name = p2.name) AS count;

 count
--------
 104846
 (1 row)
```

(b)
```
SELECT COUNT(*) FROM (SELECT DISTINCT m1.mid, m2.mid
FROM movies m1, movies m2, play_in p1,
play_in p2, play_in p3, play_in p4
WHERE m1.mid > m2.mid and m1.mid = p1.mid and m2.mid = p2.mid and
m1.mid = p3.mid and m2.mid = p4.mid and  p2.name <> p4.name
and p1.name = p2.name and p3.name = p4.name) AS count;

  count
-------
 6845
(1 row)
```

Grading info: *Graded by Ming Zhong*

- *-1 for not showing the result or the result is wrong but the query is correct.*
- *-3 for not using distinct, thus counting same result tuple more than once*
- *0 point for the subquestion if the query and answer are both not correct*

Question 8: Skyline query ..... [25 points]

Submit on separate page

(a) **[25 points]** We want to find a set of movies that have both high popularity (ie, high `num_ratings`) as well as high quality (`rating`). No single movie may achieve both - in which case, we want the so-called *Skyline* query[2]. More specifically, we want all movies that are not "dominated" by any other movie:

Definition of domination : *Movie "A"* dominates *movie "B" if movie "A" wins over movie "B", on both criteria, or wins on one, and ties on the rest.*

Figure 1 gives a pictorial example: the solid dots ('A', 'D', 'F') are not dominated by any other dot, and thus form the skyline. All other dots are dominated by at least one other dot: e.g., dot 'B' is dominated by dot 'A', being inside the shaded rectangle that has 'A' as the upper-right corner.

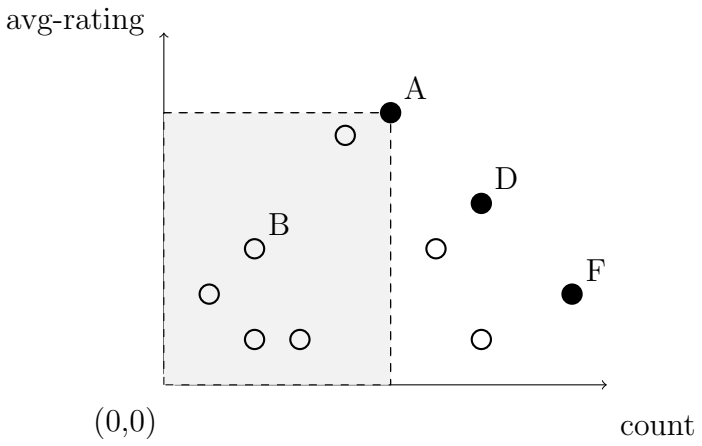

Figure 1: Illustration of Skyline and domination : 'A' dominates all points in the shaded rectangle; 'A', 'D' and 'F' form the skyline of this cloud of points

Given the above description, print the title of all the movies on the skyline, along with the rating and the number of ratings.

Solution:

```
DROP VIEW IF EXISTS Dominated;

CREATE VIEW Dominated AS
```

[2]FYI, this is also related to the *multi-objective optimization*, since we want movies that optimize two different criteria, at the same time.

```
SELECT DISTINCT m2.mid, m2.title,m2.num_ratings, m2.rating
FROM movies m1, movies m2
WHERE m2.rating<=m1.rating and m2.num_ratings<=m1.num_ratings
and NOT (m2.rating = m1.rating and m2.num_ratings=m1.num_ratings);

SELECT title,num_ratings,rating FROM movies
EXCEPT
(SELECT title,num_ratings,rating  FROM Dominated);

                  title                   | num_ratings | rating
------------------------------------------+-------------+--------
 Pirates of the Caribbean: At World's End  |     1768593 |    7.6
 The Dark Knight                           |      399618 |      9
 Harry Potter and the Order of the Phoenix |     1449179 |    7.8
 The Bourne Ultimatum                      |     1312487 |    8.2
(4 rows)
```

Grading info: *Graded by Ming Zhong*

- *-1 for not showing result/result is not correct but query is correct*
- *-3 for not considering the condition where movie1's rating equals to movie2's but movie1's num ratings are larger*
- *-2 for not considering two movies that have the same rating and num ratings*
- *-1 for not including correct query in the question*