# CMU - SCS <br> 15-415/15-615 Database Applications Spring 2013, C. Faloutsos Homework 2: SQL queries - Solutions Released: Tuesday, 02/05/2013 Deadline: Tuesday, 02/12/2013 

## Reminders - IMPORTANT:

- Like all homework, it has to be done individually.
- Please submit a printout of both your SQL commands and the output of the queries in a hard copy, in class, on Tuesday, 02/12/2013, 1:30pm.
- Further, submit your code electronically, as described in details below, by $1: 30 \mathrm{pm}$ on Tuesday, 02/12/2013.
- For ease of grading, please print each of the eight questions on a separate page, i.e., eight pages in total for this homework. If you need more pages for one problem, please staple them together. Type your name and andrew ID on each of the eight pages. FYI, we will have eight labeled piles at the front of the classroom, one for each problem.


## Reminders - FYI:

- Weight: $15 \%$ of homework grade.
- The points of this homework add up to 100 .
- Rough time estimates: 3-6 hours.


## SQL queries on the MovieLens dataset (100 points) [Kate]

In this homework we will use the MovieLens dataset released in May 2011. For more details, please refer to:
http://www.cs.cmu.edu/~christos/courses/dbms.S13/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, $t i t l e$ 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 a 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_positions of Leonardo DiCaprio and Kate Winslet are 1 and 2, respectively).

## Queries

Write SQL queries for the following:
Q 1. Print all movie titles starring Daniel Craig, sorted in ascending alpha order. [5 points]

## * SOLUTION:

select movies.title
from movies, play_in
where play_in.name='Daniel」Craig' and play_in.mid $=$ movies.mid order by movies.title ASC;
title

```
Casino Royale
Elizabeth
Infamous
Lara Croft: Tomb Raider
Layer Cake
Munich
Quantum of Solace
Renaissance
```

Road to Perdition
The Golden Compass
The Invasion
The Jacket
(12 rows)
Q 2. Print names of the cast of the movie "The Dark Knight" in ascending alpha order. [5 points]
$\star$ SOLUTION:

```
select play_in.name
from movies, play_in
where movies.title='The\lrcornerDark」Knight' and
play_in.mid = movies.mid
order by play_in.name ASC;
```

name
Aaron Eckhart
Adam Kalesperis
Aidan Feore
Andrew Bicknell
Andy Luther
Anthony Michael Hall
Ariyon Bakare
Beatrice Rosen
Bill Smille
Brandon Lambdin
Bronson Webb
Chin Han
Christian Bale
Chucky Venn
Cillian Murphy
Colin McFarlane
Craig Heaney
Crhis Perschler
Dale RIvera
Daryl Satcher
David Dastmalchian
Doug Ballard
Edison Chen

```
    Eric Roberts
    Erik Hellman
    Gary Oldman
    Gertrude Kyles
    Grahame Edwards
    Greg Beam
    Hannah Gunn
    Heath Ledger
    Helene Wilson
    Ian Pirie
    James Farruggio
    James Fierro
    James Scales
    Jennifer Knox
    Jonathan Ryland
    Joseph Luis Caballero
    Joshua Harto
    Joshua Rollins
    Keith Kupferer
    K. Todd Freeman
    Lanny Lutz
    Lateef Lovejoy
    Lorna Gayle
    Maggie Gyllenhaal
    Matthew Leitch
    Matthew O'Neill
Hatt 」Rippy
_Matt s Shallenberger
\checkmarkMelinda 」McGraw
_Michael_Andrew _Gorman
_Michael_Caine
\hookrightarrowMichael Corey FFoster
\PerpMichael ЈJai &White
Michael Stoyanov
Michael\_Vieau
_Monique _Curnen
_Morgan_Freeman
~Nancy _Crane
~Nathan }\checkmark\mathrm{ Gamble
~Nestor COrbonell
~Nigel_Carrington
```

$\lrcorner$ Nydia ${ }_{\lrcorner}$Rodriguez $\quad$ Terracina
$\lrcorner$ Olumiji Olawumi
－Patrick Clear
$\lrcorner$ Paul」Birchard
${ }_{\lrcorner}$Peter $\quad$ DeFaria
－Philip $\quad$ Bulcock
$\lrcorner$ Richard $\smile$ Dillane

- Ritchie $\quad$ Coster
－Roger $\lrcorner$ Monk
$\lrcorner$ Ronan $\lrcorner$ Summers
－Ron $\quad$ Dean
＿Sam」Derence
$\lrcorner$ Sarah J Jayne ${ }^{-}$Dunn
$\lrcorner$ Sophia $\llcorner$ Hinshelwood
$\lrcorner$ Thomas $\quad$ Gaitsch
$\llcorner$ Thomas $\lrcorner$ McElroy
」Tommy＿Campbell
＿Tommy」＇Tiny ，」Lister
$\lrcorner$ Tristan $\quad$ Tait
－Vincent $\quad$ Riotta
－Vincenzo - Nicoli
－Wai $\quad$ Wong
－W Walter $\_$Lewis
」William Armstrong
$\lrcorner$ William $\quad$ Fichtner
－Will ¿Zahrn
－Winston $\smile$ E11is
（91 rows）

Q 3. One would expect that the movie with the highest number of user ratings is either the highest rated movie or perhaps the lowest rated movie. Let's find out if this is the case here. [10 points]
3.1. Print all information (mid, title, year, num ratings, rating) for the movie with the most number of ratings (return only the top one result: there is only one in first place). [2 points]
$\star$ SOLUTION:
select *
from movies
where num_ratings $>=$ all (select max (num_ratings) from movies );

```
    mid | title | year |
4201 | Pirates of the Caribbean: At World's End | 2007 |
num_ratings | rating
-------------+--------
    1768593 | 7.6
```

3.2. Print all information (mid, title, year, num_ratings, rating) for the movie(s) with the highest rating (include all that tie for first place). Order by ascending mid. [2 points]
$\star$ SOLUTION:

```
select *
from movies
where rating >= all(select max(rating) from movies);
```



```
(4 rows)
```

3.3. Is the movie with the most number of user ratings among these highest rated movies? Print the output of the query that will check our conjecture, i.e. it will print the movies that have both: a) the highest number of ratings; and b) the highest average rating. [2 points]
$\star$ SOLUTION:

No - the movie with the most number of user ratings is not among the highest rated movies.

```
select * from
    (select * from movies where num_ratings
    >= all(select max(num_ratings) from movies)
                intersect
    (select * from movies
    where rating >= all(select max(rating) from movies)
                union
    select * from movies
    where rating <= all(select min(rating) from movies)) )
    as temp_table;
    mid | title | year | num_ratings | rating
-----+-------+------+-------------+---------
(0 rows)
```

3.4. Print all information (mid, title, year, num_ratings, rating) for the movie(s) with the lowest rating, ordered by ascending mid. [2 points]

## $\star$ SOLUTION:

select *
from movies where rating $<=$ all (select min(rating) from movies);

3.5. Is the movie with the most number of user ratings among these lowest rated movies? Print the output of the query that will check our conjecture, i.e. it will print the movies that have both: a) the highest number of ratings; and b) the lowest average rating. [2 points]
$\star$ SOLUTION:

No - the movie with the most number of user ratings is not among the lowest rated movies.

```
select * from
(select * from movies
    where num_ratings >= all(select max(num_ratings)
        from movies)
    intersect
(select * from movies
    where rating >= all(select max(rating) from movies)
    union
select * from movies
    where rating <= all(select min(rating) from movies)) )
as temp_table;
    mid | title | year | num_ratings | rating
(0 rows)
```

Q 4. Let's find out the movies with extreme ratings for the last 8 years. We split this question into the following steps. [13 points]
4.1. Print the movie year, title and rating of the lowest rated movie for each years in the period 2005-present, inclusive, in ascending year order. In case of a tie, print them all in ascending alpha order on title. [5 points]

## Hints:

1. You might want to create a view.
2. For example, the list of the worst rated movies for the years 2003-2005 is:

| year | title | rating |
| :---: | :---: | :---: |
| $------+------------------------------~$ | 3.8 |  |
| 2003 | House of the Dead | 3.8 |
| 2004 | Catwoman | 4.4 |
| 2005 | Alone in the Dark | 4.2 |

## $\star$ SOLUTION:

create view minrate_movies as
select m.year, m. title, m.rating
from movies $m$, (select year, min(rating) as minrate
from movies group by year) a
where m.year=a.year and a.minrate $=m$. rating and m. year $>=2005$
order by m.year ASC, m.title ASC;
select $*$ from minrate_movies;

| year | title | rating |
| :---: | :---: | :---: |
| 2005 | Alone in the Dark | 4.2 |
| 2006 | Basic Instinct 2 | 5 |
| 2006 | Bug | 5 |
| 2006 | Doogal | 5 |
| 2007 | D-War | 4.6 |
| 2008 | Disaster Movie | 4.6 |
| 2009 | Miss March | 5.4 |
| 2009 | The Last House on the Left | 5.4 |
| 8 row |  |  |

4.2. Print the movie year, title and rating of the highest rated movie for each years in the period 2005-present, inclusive, in ascending year order. In case of a tie, print all, sorted in ascending alpha order on the title. [5 points]
Hints:

1. You might want to create a view.
2. For example, the list of the highest rated movies for the years 2003-2005 is:

| year | title | rating |
| :---: | :---: | :---: |
| 2003 | Oldeuboi | 8.6 |
| 2004 | Bin-jip | 8.6 |
| 2005 | Chinjeolhan geumjassi | 8 |
| 2005 | Diary of a Mad Black Woman | 8 |

## $\star$ SOLUTION:

create view maxrate movies as
select m.year, m. title, m. rating
from movies m, (select year, max(rating) as maxrate
from movies group by year) a
where m. year=a.year and a.maxrate $=m$. rating and m. year $>=2005$ order by m. year ASC, m.title ASC; select $*$ from maxrate movies;

| year | title | rating |
| :---: | :---: | :---: |
| -005 | Chinjeolhan geumjassi | 8 |
| 2005 | Diary of a Mad Black Woman | 8 |
| 2005 | Das Leben der Anderen | 8.8 |
| 2007 | Into the Wild | 8.4 |


| 2007 | Persepolis | 8.4 |
| :--- | :--- | ---: |
| 2008 \| The Dark Knight | \| Star Trek |  |
| 2009 \| Sta |  |  |
| $(7$ rows $)$ |  |  |

4.3. Print the combined output (movie year, title and rating), for both the lowest and highest rated movies per year from 2005 to present, in ascending year order. As before, break ties by printing them in ascending alpha order on the title. [3 points]

## $\star$ SOLUTION:

```
select *
from (select * from minrate_movies
union select * from maxrate_movies) as best_worst;
```



Q 5. Let's find out who are the "no flop" actors: we will define a "no flop" actor as one who has played only in movies which have a rating greater than or equal to 8 . We split this problem into the following steps. [12 points]
5.1. Create a view called high_ratings which contains the distinct names of all actors who have played in movies with a rating greater than or equal to 8 . Similarly, create a view called low_ratings which contains the distinct names of all actors who have played in movies with a rating less than 8 . Print a) the number of rows in the view high_ratings and $b$ ) the number of rows in the view low_ratings. [3 points]
$\star$ SOLUTION:

```
create view high_ratings as select distinct(play_in.name)
    from play_in, movies
where play_in.mid = movies.mid and movies.rating >=8;
create view low_ratings as select distinct(play_in.name)
    from play_in, movies
where play_in.mid = movies.mid and movies.rating <8;
select count(*) from high_ratings;
    count
    5 6 6 6
(1 row)
select count(*) from low_ratings;
    count
    39930
(1 row)
```

5.2. Use the above views to print the number of "no flop" actors in the database. [2 points]
$\star$ SOLUTION:

```
select count(*) from (select * from high_ratings
    except
select * from low_ratings) as all_no_flop;
```

    count
    -------
3610
(1 row)
5.3. Finally, use the above view to print the names of these "no flop" actors, along with the number $M$ of movies they have played in, sorted by descending $M$ and then by ascending name, and print only the top 10. [7 points]

## $\star$ SOLUTION:

create view no_flop as select * from
(select $*$ from high_ratings except select $*$ from low_ratings) as all_no_flop;

```
select no_flop.name, count(movies.mid) as M
                        from no_flop, movies, play_in where
    movies.mid = play_in.mid and play_in.name=no_flop.name
    group by no_flop.name order by M desc, no_flop.name asc
    limit 10;
```



Q 6. Let's find out who is the actor with the highest "longevity." Print the name of the actor/actress who has been playing in movies for the longest period of time (i.e. the time interval between their first movie and their last movie is the greatest). [15 points]

## $\star$ SOLUTION:

create view helper as
(select name, ( $\max ($ year $)-\min (y e a r))$ as diff
from movies, play_in
where movies.mid = play_in.mid
group by name order by 2 DESC);
select helper. name
from helper where helper.diff $>=$
all (select helper.diff from helper)
order by helper. name asc;
name

```
Edward de Souza
John Normington
(2 rows)
```

Q 7. Let's find the close buddies of Annette Nicole: print the names of all actors who have starred in all movies in which Annette Nicole has starred in (it's ok to report the name of Annette Nicole in the result; also, it is ok if these actors have starred in more movies than Annette Nicole has played in). PostgreSQL does not have a relational division operator, so we will guide you through the following steps (you might find it useful to consult the slides or the textbook for the alternative "double negation" method of performing relational division). [15 points]
7.1. First, create a view co_actors which contains the distinct names of actors who have played in at least one movie with Annette Nicole. Print the number of rows in this view. [3 points]
$\star$ SOLUTION:

```
create view co_actors as
        select distinct(name) from
        (select m.mid from movies m, play_in a
        where a.name='Annette}\downarrowNicole' and a.mid=m.mid
        as an_movies, play_in a2
        where a2.mid = an_movies.mid;
select count(*) from co_actors;
    count
-------
    178
(1 row)
```

7.2. Second, create a view all_combinations which contains all possible combinations of co_actors and the movie ids in which Annette Nicole has played in. This view should have two columns: co-actor names and movie ids (note: this view contains co_actor, mid combinations which are fake, because they never happened!). Print the number of rows in this view. [4 points]
$\star$ SOLUTION:
create view all_nicole as select mid
from play_in where name $=$ 'Annette ${ }_{\lrcorner}$Nicole';
create view all_combinations as
select * from co_actors, all_nicole;
select count (*) from all_combinations;

```
    count
```

    --------
    356
    (1 row)
7.3. Third, create a view non_existent from the view all_combinations by removing all (co_actor, mid) pairs that did in fact happen (i.e. these pairs exist in the table play_in). Print the number of rows in this view. [4 points]

## $\star$ SOLUTION:

```
create view non_existent as select *
        from all_combinations
        except
        (select name, mid from play_in);
select count(*) from non_existent;
```

    count
    --------
175
(1 row)
7.4. Finally, from all of Annette Nicole's co-actors (view co_actors) remove the distinct names of the actors from the non_existent view. Print the names of the actors who have actually played in (at least) all movies that Annette Nicole has played in (note: it is ok to report Annette Nicole in the results).[4 points]

## $\star$ SOLUTION:

```
select * from co_actors
    except
(select distinct(name) from non_existent);
    name
```

```
Annette Nicole
```

Annette Nicole
Christian Perry
Christian Perry
Kristen Connolly
Kristen Connolly
(3 rows)

```
(3 rows)
```

Q 8. Let's find out who is the most "networked" actor - the actor who has the highest number of distinct co-actors.

To help you with partial credit, we split this question into the following steps. [25 points]
8.1. First, for a warm-up, print all of Jim Parsons' distinct co-actors, ordered by ascending co-actor name. We will call Jim Parsons the "source actor." ${ }^{1}[10$ points $]$

Hint: For example, the distinct co-actors of Rupert Murray are:

```
    src_name | co_actor_name
---------------+-----------------
Rupert Murray | Daniel Schacter
Rupert Murray | Doug Bruce
Rupert Murray | Douglas Bruce
```

$\star$ SOLUTION:
select p1.name, p 2. name as coactors_N
from play_in p1, play_in p2
where $\mathrm{p} 1 . \mathrm{mid}=\mathrm{p} 2 . \mathrm{mid}$ and p 1. name $={ }^{\prime} \mathrm{Jim}_{\sqcup} \mathrm{Parsons}$,
group by p 1. name, p 2 . name order by 2 asc;

| name | coactors_n |
| :---: | :---: |
| Jim Parsons | Alexandra Berardi |
| Jim Parsons | Alex Burns |
| Jim Parsons | Amy Ferguson |
| Jim Parsons | Ann Dowd |
| Jim Parsons | Anne Dudek |
| Jim Parsons | Armando Riesco |
| Jim Parsons | Ato Essandoh |
| Jim Parsons | Aunjanue Ellis |
| Jim Parsons | Bobby Cannavale |
| Jim Parsons | Christopher Carley |
| Jim Parsons | Danny DeVito |
| Jim Parsons | Denis O'Hare |
| Jim Parsons | Geoffrey Arend |
| Jim Parsons | George C Wolfe |
| Jim Parsons | Ian Holm |
| Jim Parsons | Jackie Hoffman |
| Jim Parsons | Jayne Houdyshell |
| Jim Parsons | Jean Smart |
| Jim Parsons | Jennifer Echols |
| Jim Parsons | Jim Parsons |

[^0]```
Jim Parsons | Jonah Hill
Jim Parsons | Kenneth Graymez
Jim Parsons | Kumar Pallana
Jim Parsons | Leonardo Nam
Jim Parsons | Method Man
Jim Parsons | Michael Weston
Jim Parsons | Morgan Freeman
Jim Parsons | Natalie Portman
Jim Parsons | Paz Vega
Jim Parsons | Peter Sarsgaard
Jim Parsons | Rhea Perlman
Jim Parsons | Ron Leibman
Jim Parsons | Zach Braff
(33 rows)
```

8.2. Now, print only the name of the source actor (Jim Parsons) and the count of all of his distinct co-actors. [5 points]

Hint: in our previous example for Rupert Murray, this would be:

## $\star$ SOLUTION:


select p1.name, count(distinct(p2.name)) as coactors_N
from play_in p 1 , play_in p 2
where $\mathrm{p} 1 . \mathrm{mid}=\mathrm{p} 2 . \mathrm{mid}$ and p 1. name $=$ 'Jim」Parsons,
group by p 1. name;

```
    name | coactors_n
-------------+-------------
    Jim Parsons | 33
(1 row)
```

8.3. Finally, use the above query to get the distinct count of everyone's co-actors and print the (source) actor with the highest such count. The output should have two columns: the (source) actor name and the count of his distinct co-actors. In case of a tie, print all of them, sorted in alpha order on the source name. [10 points]

## $\star$ SOLUTION:

```
create view popular as
select p1.name as src_name, count(distinct(p2.name))
as coactors_N
from play_in p1, play_in p2
where p1.mid = p2.mid and p1.name <> p2.name
group by p1.name order by 2 DESC;
select src_name, coactors_N
from popular
where coactors_N >= all( select max(coactors_N)
from popular) limit 5;
src_name | coactors_N
----------------+-----------
    Robert De Niro | 1279
```


[^0]:    ${ }^{1}$ In IMDB, he is an emerging star of 2012 and a two-time Emmy-winning star of "The Big Bang Theory."

