SQL

Hasan Abuelreich

November 30, 2015

### For most problems the large data set 8gb is used. The most recent dataset 1gb is used for a couple of problems and it will be stated at the begininng of the problem when it is used.

#### I used the SQL tutorial site posted by micheal on piazza to aid me in all of the following solutions. It provided almost all the necessary information and principles needed for the SQL part of this report. Beginning with number 1, I realized that I will be needing to look at the columns of tables with a LIMIT many times, hence I created a function in R to make this simpler for all the problems. To answer this question, I looked at role\_type in order to distinguis the actor and actresses. The actor and actress roleid was 1 and 2 respecitively. Hence, I used the castinfo table to count the distinct personid for all actors and actresses by conditioning on their roleid. The total actors and actresses in this set found was 3,492,018.

library(RSQLite)

## Loading required package: DBI

library(DBI)  
setwd("~/Desktop")  
db = dbConnect(drv = SQLite(), dbname = 'lean\_imdbpy.db')  
#I created an R function that will alow me to simply view the first 10 rows for all columns for a certain table instead of having to write it in SQL each time  
#most of the headdb(),dbListTables,etc. commands were removed from the report in order not to print out many tables   
headdb = function(table){  
 dbGetQuery(db, paste0('SELECT \* FROM ', table, ' LIMIT 10'))  
}  
dbListTables(db)

## [1] "aka\_name" "aka\_title" "cast\_info"   
## [4] "info\_type" "keyword" "kind\_type"   
## [7] "movie\_info" "movie\_info\_idx" "movie\_keyword"   
## [10] "name" "person\_info" "role\_type"   
## [13] "sqlite\_sequence" "title"

headdb('cast\_info')

## id person\_id movie\_id person\_role\_id note nr\_order role\_id  
## 1 1 1 1402868 1 <NA> NA 1  
## 2 2 2 3049788 1 <NA> 25 1  
## 3 3 2 3259788 2 <NA> 22 1  
## 4 4 3 2647137 NA <NA> 12 1  
## 5 5 4 2372097 3 <NA> NA 1  
## 6 6 4 2372098 1 <NA> NA 1  
## 7 7 4 2377018 1 <NA> NA 1  
## 8 8 4 2427492 4 <NA> NA 1  
## 9 9 4 2474580 1 <NA> NA 1  
## 10 10 4 2510490 NA <NA> 7 1

#found from role\_type table, role\_id for actor/actress 1 & 2 respectively  
#person\_id is used to count   
dbGetQuery(db, 'SELECT COUNT(DISTINCT(person\_id))   
 FROM cast\_info   
 WHERE role\_id = 1 or role\_id = 2')

## COUNT(DISTINCT(person\_id))  
## 1 3492018

#### I used max and min of production year from the title to determine the time span that is covered. It seems that the data covers the time span of 1874-2025. 2025 is the max because movies that are planned or in production are not released yet and they will be released in 2024 fo example.

dbGetQuery(db, 'SELECT MAX(production\_year) FROM title')

## MAX(production\_year)  
## 1 2025

# max is 2025  
  
dbGetQuery(db, 'SELECT MIN(production\_year) FROM title')

## MIN(production\_year)  
## 1 1874

# min title appears to be 1874

#### In the following problem, I used names to count the distinct names and divided that by proportion of all names. In order to get proportion for male and female actors only, I filtered for role as actor or actress and grouped everything by gender at the end. The results are females consisting of 37% of all actors and males consisting of 63% of all actors. The following chunk would not run in markdown but R separately. That is why the code is not in a chunk and neatly aligned. I really had issues with this one and didn’t know until I was knitting.

*dbGetQuery(data, 'SELECT gender, (COUNT(DISTINCT name)\*1.0/(SELECT COUNT(DISTINCT name) FROM name, cast\_info, role\_type*

*WHERE cast\_info.role\_id = role\_type.id AND cast\_info.person\_id = name.id AND role IN ("actor", "actress"))) AS Proportions*

*FROM name, cast\_info, role\_type*

*GROUP BY gender')*

*Gender Proportions:*

*#1 f 0.3724914 = 37% female*

*#2 m 0.6342281 = 63% male*

#### For this problem, db2 is used which is the data that is posted most recently containing 10% of all the data in order to run faster. A similar appraoch was used to the previous problem in obtaining the proportions. The kindid, kind, and proportions are displayed and it is evident that the most frequent "kind" are movies consisting of 79.66% of this entire data set. Tv series, tv movie, & video game are 13.8%, 7.5%, and .95% respectively.

setwd("~/Desktop")  
db2 = dbConnect(drv = SQLite(), dbname = 'lean\_imdbpy\_2010.db')  
dbGetQuery(db2, 'SELECT kind\_id, kind, (COUNT(DISTINCT title)\*1.0/  
 (SELECT COUNT(DISTINCT title) FROM title2))   
 AS Proportions FROM title2, kind\_type   
 WHERE title2.kind\_id = kind\_type.id GROUP BY kind\_id')

## kind\_id kind Proportions  
## 1 1 movie 0.796680951  
## 2 2 tv series 0.138107129  
## 3 3 tv movie 0.075124026  
## 4 6 video game 0.009594177

#### This problem was a little challanging at first, but post 1238 on piazza was very helpful as Professor Duncan posted the manner in the ways should be connected in. After more reading on piazza I found that movie\_info possesed the info for genre. Hence, I checked infotype table and found that the infotype id for genre is 3. To find all the differen genres I used the DISTINCT function in order to get only unique genres. I joined movieinfo and infotype tables based on the common columns 'infotypeid' and 'id' respectively. I used WHERE to condition only to display genres. Because I only counted only distinct genres in the very beginning, all the distinct genres are displayed below in the table.

headdb('info\_type')

## id info  
## 1 1 runtimes  
## 2 2 color info  
## 3 3 genres  
## 4 4 languages  
## 5 5 certificates  
## 6 6 sound mix  
## 7 7 tech info  
## 8 8 countries  
## 9 9 taglines  
## 10 10 keywords

dbGetQuery(db, 'SELECT DISTINCT(movie\_info.info)  
 FROM movie\_info  
 INNER JOIN info\_type  
 ON info\_type.id = movie\_info.info\_type\_id WHERE info\_type.id = 3')

## info  
## 1 Documentary  
## 2 Reality-TV  
## 3 Horror  
## 4 Drama  
## 5 Comedy  
## 6 Musical  
## 7 Talk-Show  
## 8 Mystery  
## 9 News  
## 10 Sport  
## 11 Sci-Fi  
## 12 Romance  
## 13 Family  
## 14 Short  
## 15 Biography  
## 16 Music  
## 17 Game-Show  
## 18 Adventure  
## 19 Crime  
## 20 War  
## 21 Fantasy  
## 22 Thriller  
## 23 Animation  
## 24 Action  
## 25 History  
## 26 Adult  
## 27 Western  
## 28 Lifestyle  
## 29 Film-Noir  
## 30 Experimental  
## 31 Commercial  
## 32 Erotica

# 32 different genres; names are the following:

#### In order to obtain the 10 most common genres, GROUP BY and ORDER BY are used. I used the SQL code for the previous problem because it already contained the genres, but modified the code in order to obtain the 10 most common genres. I grouped the results to this code by genres by using the GROUP BY and following that I order the results by count. In order to get the ordering desired, I added "desc" after the count and limited it to 10 results in order to obtain the top 10 genres displayed below.

#groups by genres & orders it from most to least  
#remove count to view genres & add count to view there counts  
  
dbGetQuery(db, 'SELECT DISTINCT(movie\_info.info)  
 FROM movie\_info  
 INNER JOIN info\_type  
 ON info\_type.id = movie\_info.info\_type\_id WHERE info\_type.id = 3  
 GROUP BY movie\_info.info   
 ORDER BY COUNT(movie\_info.info) desc LIMIT 10')

## info  
## 1 Short  
## 2 Drama  
## 3 Comedy  
## 4 Documentary  
## 5 Adult  
## 6 Action  
## 7 Romance  
## 8 Thriller  
## 9 Animation  
## 10 Family

#### This problem was a challanging one for me, but after reading numerous posts on piazza about the keword I decided to look for any keyword containing the word "space". I joined keyword, moviekeyword, and title tables in order to to have all the necessary tables to obtain the keyword "space". I used %space% in order to count space coming after a word or before. LIKE was used to match %space% because it seeemed to me that it was the regular expression form in SQL. The total number movies containing the word "space" are 392.

dbGetQuery(db, 'SELECT COUNT(keyword.keyword), movie\_keyword.movie\_id, title.title, title.production\_year  
 FROM keyword   
 JOIN movie\_keyword  
 ON keyword.id = movie\_keyword.id   
 JOIN title  
 ON title.id = movie\_keyword.id   
 WHERE keyword.keyword LIKE "%space%" LIMIT 5')

## COUNT(keyword.keyword) movie\_id title production\_year  
## 1 392 272917 Jak vyuzít promeny 2006

#### In order to obtain years released, and the top five actors in these movies I modified the code above to include the cast\_info to obtain nrorder and also in order to connnect the name table because they have a mutual column. It seemed to me that is the simplest way to connect the actors name column to our previous tables. I used nrorder column to classify the lead actors because from what I understood in office hours, nrorder classifies the lead actors and starts at 1 for lead and increases onwards. I condition nrorder to being between 0 and 5, so this way the top 5 actors will be displayed for each movie. I limited the output to 10 for the analysis to display a sample of output.

#nr\_order is used to classify the top actors and   
#it is filtered from 1-5 to include the top 5 actors for each movie containing the keyword "space"  
headdb('cast\_info')

## id person\_id movie\_id person\_role\_id note nr\_order role\_id  
## 1 1 1 1402868 1 <NA> NA 1  
## 2 2 2 3049788 1 <NA> 25 1  
## 3 3 2 3259788 2 <NA> 22 1  
## 4 4 3 2647137 NA <NA> 12 1  
## 5 5 4 2372097 3 <NA> NA 1  
## 6 6 4 2372098 1 <NA> NA 1  
## 7 7 4 2377018 1 <NA> NA 1  
## 8 8 4 2427492 4 <NA> NA 1  
## 9 9 4 2474580 1 <NA> NA 1  
## 10 10 4 2510490 NA <NA> 7 1

headdb('name')

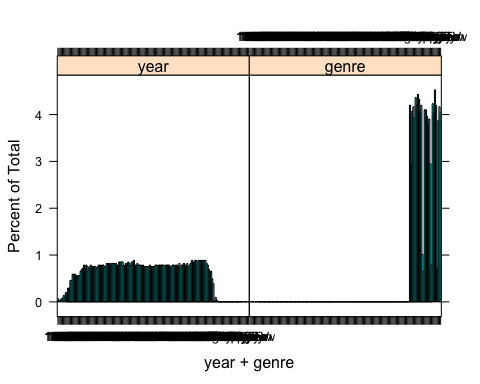
## id name imdb\_index imdb\_id gender  
## 1 1 $, Claw <NA> NA m  
## 2 2 $, Homo <NA> NA m  
## 3 3 $, Steve <NA> NA m  
## 4 4 $hort, Too <NA> NA m  
## 5 5 $lim, Bee Moe <NA> NA m  
## 6 6 $ly, Yung <NA> NA m  
## 7 7 $torm, Cuntry <NA> NA m  
## 8 8 & Davi, Bruninho <NA> NA m  
## 9 9 & Dollar Furado, Caio Corsalette <NA> NA m  
## 10 10 & Fabiano, César Menotti <NA> NA m  
## name\_pcode\_cf name\_pcode\_nf surname\_pcode  
## 1 C4 <NA> <NA>  
## 2 H5 <NA> <NA>  
## 3 S31 <NA> <NA>  
## 4 H63 T63 <NA>  
## 5 L515 B545 L5  
## 6 L52 Y524 L  
## 7 T6525 C5363 T65  
## 8 D165 B6531 D1  
## 9 D4616 C2624 <NA>  
## 10 F1526 C2653 F15  
## md5sum  
## 1 1ea1ae4a123a67b1939650abd92773bb  
## 2 42b7768f2a5b112fb2f8c5a3c7cbebdd  
## 3 776f400962ab7b9a07976bcafa31d8fa  
## 4 d17acd6c48e9d76dc6cf80e6ba1ca0f9  
## 5 b3b45ecf9ed9e65ba1a2ce2cfd19f9c3  
## 6 80ed7effa280cb0e013f4300d0b0421d  
## 7 6e146b62c39ec4671a178c5865ef06c9  
## 8 9f2a433e2c21ecc51da0ab3238ee4809  
## 9 379e2ec3525c8a0746825ce35b3d89c6  
## 10 6d205a291e82fe07344c68a5282916e5

dbGetQuery(db, 'SELECT keyword.keyword, movie\_keyword.movie\_id, title.title, title.production\_year, cast\_info.nr\_order, name.name  
 FROM keyword   
 JOIN movie\_keyword  
 ON keyword.id = movie\_keyword.id   
 JOIN title  
 ON title.id = movie\_keyword.id   
 JOIN cast\_info  
 ON cast\_info.movie\_id = movie\_keyword.movie\_id  
 JOIN name  
 ON cast\_info.person\_id = name.id  
 WHERE keyword.keyword LIKE "%space%"   
 AND nr\_order < 5  
 AND 0 < nr\_order LIMIT 10')

## keyword movie\_id title production\_year nr\_order  
## 1 space-camp 2842 + Clair 2001 1  
## 2 space-camp 2842 + Clair 2001 4  
## 3 space-camp 2842 + Clair 2001 3  
## 4 space-camp 2842 + Clair 2001 2  
## 5 space-shuttle 7076 1 Girl 5 Gays 500 Fans 2011 2  
## 6 space-shuttle 7076 1 Girl 5 Gays 500 Fans 2011 4  
## 7 space-shuttle 7076 1 Girl 5 Gays 500 Fans 2011 1  
## 8 space-shuttle 7076 1 Girl 5 Gays 500 Fans 2011 3  
## 9 space-travel 13092 Een bocht teveel 1995 1  
## 10 space-travel 13092 Een bocht teveel 1995 3  
## name  
## 1 Mangan, Stephen  
## 2 Whitby, Jack  
## 3 Cruwys-Finnigan, Celia  
## 4 Tarbuck, Liza  
## 5 Craft, Rodney  
## 6 Minshew, Cody  
## 7 Nocar, Melyssa  
## 8 Paul, Laura  
## 9 Sutherland, Kiefer  
## 10 Clarke, Sarah

#### For the following problem I created a table consisting of all year, genres, and number of movies for each corresonding year. I constarint the movies to kind\_id=1 to only count movies. Following that I used the lattice package to obtain the plot.

genre\_time = dbGetQuery(db, "SELECT info, production\_year, COUNT(DISTINCT title)   
 AS Movie\_Number   
 FROM movie\_info, title  
 WHERE movie\_info.movie\_id = title.id AND kind\_id = 1 AND info\_type\_id = 3   
 GROUP BY production\_year, info")  
  
count = genre\_time$Movie\_Number  
genre = as.factor(genre\_time$info)  
year = as.numeric(genre\_time$production\_year)  
library('lattice')  
histogram(count ~ year + genre)



#### In order to obtain the actors that have been in the most movies, I joined castinfo, name, roletype, and title tables. This way I can condition on actor/actress, make sure only movies are counted using their unique kindid, and count the most occuring names by grouping the names and sorting by descinding order. The output of the top 20 actors is displayed below.

dbGetQuery(db, "SELECT cast\_info.person\_id, name.name, role\_type.role, title.kind\_id  
 FROM cast\_info  
 JOIN name, role\_type, title  
 ON cast\_info.person\_id=name.id AND cast\_info.role\_id=role\_type.id AND title.id = cast\_info.movie\_id   
 WHERE role\_id = 1 AND kind\_id = 1  
 OR role\_id = 2 AND kind\_id = 1  
 GROUP BY name.name  
 ORDER BY COUNT (name.name) DESC  
 LIMIT 20")

## person\_id name role kind\_id  
## 1 195959 Blanc, Mel actor 1  
## 2 233482 Brahmanandam actor 1  
## 3 1509290 Onoe, Matsunosuke actor 1  
## 4 2615064 Flowers, Bess actress 1  
## 5 1344110 Mercer, Jack actor 1  
## 6 801783 Hack, Herman actor 1  
## 7 831933 Harris, Sam actor 1  
## 8 3134277 Phelps, Lee actress 1  
## 9 969854 Jeremy, Ron actor 1  
## 10 1016068 Kapoor, Shakti actor 1  
## 11 382758 Cobb, Edmund actor 1  
## 12 1488005 O'Connor, Frank actor 1  
## 13 1362635 Miller, Harold actor 1  
## 14 1199644 London, Tom actor 1  
## 15 1414125 Mower, Jack actor 1  
## 16 611864 Farnum, Franklyn actor 1  
## 17 1516543 Osborne, Bud actor 1  
## 18 1693202 Richardson, Jack actor 1  
## 19 695727 Garcia, Eddie actor 1  
## 20 580276 Ellis, Frank actor 1

#### In order to obtain the actors that had the most number of movies with "top billing", "nrorder" was used to classify top billing and I chose to distinguish the top 3 actors in each movie as the top billing ones. In order to show the years these movies spanned I also needed the production year, hence I joined castinfo, name, roletype, and title tables to get all the desired columns and be able to condition only for roles of actors and obtain th production year from the title. Name and personid are obviously needed here to identify the person. I condition my nroder to be between 0 and 3 that way the result will be only the top 3 actors for each movie. Roleid is set less than 3 to only include actor and actress. The output is limited to 5 in order to display a sample of the results.

dbGetQuery(db, "SELECT DISTINCT cast\_info.person\_id, name.name,cast\_info.nr\_order, role\_type.role, title.production\_year,COUNT(cast\_info.movie\_id)  
 FROM cast\_info INNER JOIN name, role\_type, title  
 ON cast\_info.person\_id=name.id AND cast\_info.role\_id=role\_type.id AND title.id = cast\_info.movie\_id  
 WHERE nr\_order > 0 AND nr\_order < 3 AND role\_id < 3  
 GROUP BY name  
 ORDER BY COUNT(movie\_id) DESC  
 LIMIT 5")

## person\_id name nr\_order role production\_year  
## 1 228359 Boyd, Jim 2 actor 1971  
## 2 3426334 Vorderman, Carol 1 actress 2007  
## 3 2174700 Whiteley, Richard 1 actor 1999  
## 4 1723810 Rogers, Fred 1 actor 1963  
## 5 326223 Castellaneta, Dan 1 actor 1996  
## COUNT(cast\_info.movie\_id)  
## 1 6241  
## 2 4651  
## 3 4511  
## 4 4411  
## 5 3650

#### The smaller most recent dataset posted on piazza was used here. In order to obtain the 10 actors that performed in the most movies within any given year, I needed the name, personid,title, and kindid columns. Hence, I joined the name2, castinfo2, and title2 tables. I used the "kindid"" to filter only for movies. In this case personid from castinfo was used in order to make a connection from the name2 table to the title2 table. I filtered for actor and actress roles using "roleid" and grouped the results by count of name and in descinding order. I limited the results to 10 in order to get the top 10 actors. The results are the following table.

dbGetQuery(db2, 'SELECT name2.name, cast\_info2.person\_id, title2.title, title2.production\_year, title2.kind\_id  
 FROM name2  
 JOIN cast\_info2  
 ON cast\_info2.person\_id = name2.id  
 JOIN title2  
 ON title2.id = cast\_info2.movie\_id   
 WHERE role\_id = 1 AND kind\_id = 1  
 OR role\_id = 2 AND kind\_id = 1  
 GROUP BY name2.name  
 ORDER BY COUNT(name2.name) DESC LIMIT 10')

## name person\_id title  
## 1 Roberts, Eric 1708796 An Act of Malice  
## 2 Kaufman, Lloyd 1025363 Évocateur: The Morton Downey Jr. Movie  
## 3 O'Connor, George 1488023 Writer's Block  
## 4 Brahmanandam 233482 Yevadu  
## 5 Lorente, Txema 1204854 Zona de Caza  
## 6 Pell, Rhoda 3123037 Woggie  
## 7 Rivers, Scott 1705106 Yellow Wasp  
## 8 Thingvall, Joel 2012005 Zombie Homecoming  
## 9 Oberst Jr., Bill 1494693 The Border  
## 10 Olsen, Maria 3091849 William Froste  
## production\_year kind\_id  
## 1 2015 1  
## 2 2012 1  
## 3 2015 1  
## 4 2014 1  
## 5 2012 1  
## 6 2012 1  
## 7 2015 1  
## 8 2010 1  
## 9 2015 1  
## 10 2015 1

#### In order to obtain the 10 actors with the most aliases, the akanames table needs to be joined onto castinfo and name tables. We need the akanames table and names to match the names column in akanames with the names column in names table. Castinfo table will be used in order to filter for roleid to only include actor or actress. After joining the tables and making the connections between them, roles are filtered for and grouping is done by names and the count is for the aliases. Hence, now by grouping in DESC order, the most occuring to least will appear, which is the desired result. Limiting it to the top 20 results in the following table.

# by looking at aka\_name I noticed that the person\_id in this table was repeated for each name so I count the person\_id and ordered it by it  
 # then I put role\_id<3 for actors and actress and group by name to look at the names  
 dbGetQuery(db, "SELECT DISTINCT aka\_name.person\_id, name.name, cast\_info.role\_id,COUNT(aka\_name.person\_id)  
 FROM aka\_name INNER JOIN name, cast\_info  
 ON cast\_info.person\_id=name.id AND name.id=aka\_name.person\_id  
 WHERE role\_id<3  
 GROUP BY name.name  
 ORDER BY COUNT(aka\_name.person\_id)DESC  
 LIMIT 20")

## person\_id name role\_id COUNT(aka\_name.person\_id)  
## 1 326223 Castellaneta, Dan 1 125523  
## 2 97822 Azaria, Hank 1 91329  
## 3 1848752 Shearer, Harry 1 69853  
## 4 2939412 MacNeille, Tress 2 64832  
## 5 969854 Jeremy, Ron 1 59179  
## 6 1796694 Savage, Herschel 1 58353  
## 7 2161819 Welker, Frank 1 55195  
## 8 1486206 O'Brien, Conan 1 52663  
## 9 1869225 Silvera, Joey 1 48762  
## 10 2436109 Cartwright, Nancy 2 44935  
## 11 2684230 Griffin, Grey 2 42408  
## 12 2020756 Thornton, Kirk 1 41601  
## 13 201825 Blum, Steve 1 41006  
## 14 163529 Bennett, Jeff 1 38744  
## 15 1840683 Shaffer, Paul 1 36407  
## 16 109175 Baker, Dee Bradley 1 33180  
## 17 1556393 Paulsen, Rob 1 32652  
## 18 413989 Cosby, Bill 1 29436  
## 19 3430142 Wahlgren, Kari 2 28455  
## 20 1763137 Sabat, Christopher 1 24832

#### First I start out by picking George Clooney as my actor and following that I join the cast\_info, name, and title tables because they contain all the reequired information. I condition my code to only search for movies that contain George Clooney as the lead actor which would be conditioning George Clooney on nr\_order = 1 in this case. Following that I choose Ocean Thirteen as a movie and to find the supporting actors that played with George Clooney, I build on the previous code, but condition on nr\_order between 1 and and up to 3 so I can get the other actors excluding Clooney.

clooney = dbGetQuery(db, "SELECT cast\_info.movie\_id,cast\_info.nr\_order, name.name ,title.title  
 FROM cast\_info, name, title  
 WHERE cast\_info.person\_id=name.id AND cast\_info.movie\_id=title.id  
 AND name.name='Clooney, George'  
 AND cast\_info.nr\_order=1")  
  
roles = dbGetQuery(db, "SELECT movie\_id, person\_id, role\_id, nr\_order, name.name ,title.title  
 FROM cast\_info, name, title  
 WHERE cast\_info.person\_id=name.id AND cast\_info.movie\_id=title.id  
 AND nr\_order>1 AND role\_id<3 AND movie\_id = 3055221  
 GROUP BY name.name  
 ORDER BY nr\_order  
 LIMIT 100")