Tiffany Chen 11/20/15 ID: 998840686

STA 141 Assignment 5

Introduction

Learning to use SQL commands to query databases in R using a mix of SQL and R commands. We are exploring the IMDb database, which is about actors and movies and their relations.

Statement

```
I did this assignment by myself and developed and wrote the code for each part by myself,
drawing only from class, section, Piazza posts and the Web.  I did not use code from a
fellow student or a tutor or any other individual.
```

Tiffany Chen

Notes: I used the new data set with the 8 tables dropped titled lean_imdbpy.db unless explicitly mentioned otherwise. Also, if I did not explicitly put the code in the output, it will be in the appendix.

Questions

1. **Q1: How many actors are there in the database? How many movies?**
   There are 1936807 distinct actors and 722933 distinct movies. I came to these numbers by merging tables and then counting distinct names or movie titles. For the actors, I merged cast_info and name and looked only at actors which means the role_id is 1. Then, I counted the distinct column of names. For movies, I used the titles table and filtered only the movies, which is where kind_id is 1. Then, I counted the distinct column of titles.

```
dbGetQuery(db, "SELECT COUNT(DISTINCT(title))
       FROM title
       WHERE kind_id = 1;") # 722933 movies

dbGetQuery(db, "SELECT COUNT( DISTINCT(name))
      FROM cast_info, name
      WHERE role_id = 1
       AND name.id = cast_info.person_id;") # 1936807 actors
```

2. **Q2: What time period does the database cover?**
   The database covers from 1874 to 2018. I used the min and max of the production_year from the title table to determine the years. These years are not limited to just movies. I looked up the 1874 movie which is Passage de Venus and it is indeed from 1874 according to the IMDB website. It is a record of the transit of the planet Venus across the Sun. the 2018 movie includes many such as Jurassic World Sequel, Terminator 3, etc which all seems very reasonable.

```
dbGetQuery(db, "SELECT MIN(production_year)
       FROM title;") # 1874 min yr
dbGetQuery(db, "SELECT MAX(production_year)
       FROM title;") # 2025 max yr
```

3. **Q3: What proportion of the actors are female? male?**
   The proportions of the names in the dataset not just limited to actors are:

```
# 1  <NA>                    34.89949
# 2    f                  23.02708
# 3    m                    42.07343
```

I grouped the data from the names table by gender. To calculate the actual percentages, I first count up the total by gender, then I divided that by the total count. Based on a Piazza post and what was said in class, I multiplied the count by 100.0 to avoid getting 0s or 1s.

```
dbGetQuery(db, "SELECT gender,
          COUNT(*) * 100.0 / (SELECT COUNT(*) FROM name)
         FROM name GROUP BY gender;") # actual proportions
```

4. **Q4: What proportion of the entries in the movies table are actual movies and what proportion are television series, etc.?**

   The proportions of movies, etc:

```
1     episode                           63.5583712
2       movie                          24.9111894
3    tv movie                            3.4126175
4   tv series                          3.5273371
5  video game                              0.4341033
6 video movie                              4.1563815
```

I first created a new table named movie that merged the title with kind_type so I can later group by the kind_type. I followed the same strategy as the previous problem. This time, I grouped by the kind.

```
# merge kind of movies with the title table
dbGetQuery(db, "CREATE TABLE movie AS SELECT *
       FROM title AS t, kind_type AS k
         WHERE t.kind_id = k.id;")
# return proportions of entries
dbGetQuery(db, "SELECT kind, COUNT(*) * 100.0 /
          (SELECT COUNT(*) FROM movie)
         FROM movie GROUP BY kind;")
```

5. **Q5: How many genres are there? What are their names/descriptions?**

There are 32 genres.
The first five are:

```
1   Documentary
2    Reality-TV
3       Horror
4        Drama
5       Comedy
```

The notes/description are all NA. By noticing that 3 corresponds to the genres in the info_type table, I specifically looked only at those in the info_type_id column of the movie_info table. Then, I wanted the distinct names of the genres.

```
dbGetQuery(db, "SELECT DISTINCT info
         FROM movie_info
         WHERE info_type_id = 3;") # genre is when info_type_id is 3
```

6. **Q6: List the 10 most common genres of movies, showing the number of movies in each of these genres.**

   The ten most common genres of movies in the movie_info table (which might not be limited to just movies) is:

```
   info COUNT(info)
```

```
1       Short      522672
2       Drama      330171
3      Comedy      243486
4  Documentary     215477
5       Adult      71983
6      Action      62830
7     Romance      62756
8    Thriller      61789
9   Animation      54606
10      Family      52197
```
I used the same table I used from the previous problem but I group by the info then order by the total count and sorted in descending order and got the top 10 by doing LIMIT 10.

```
dbGetQuery(db, "SELECT info, COUNT(info)
           FROM movie_info
           WHERE info_type_id = 3
          GROUP BY info
          ORDER BY COUNT(*) DESC LIMIT 10;")
```

7. **Q7: Find all movies with the keyword 'space'. How many are there? What are the years these were released? and who were the top 5 actors in each of these movies?**
To find all movies with the keyword ‘space’, I interpreted that as finding all movies with keyword = ‘space’ from the keywordtitle table in the data set. I got 400 “space” movies in this way. First, I made a table that combined the keywords with the title. Then, I narrowed it down by movies so kind_id = 1 and space, so keyword = ‘space’. I counted the distinct titles and 400 was the result.

```
dbGetQuery(db, "CREATE TABLE keywordtitle AS
         SELECT * FROM title, movie_keyword, keyword
         WHERE title.id = movie_keyword.movie_id
         AND movie_keyword.keyword_id = keyword.id")

dbGetQuery(db, "CREATE TABLE spacemovies AS
           SELECT * FROM keywordtitle
           WHERE keyword = 'space' AND kind_id = 1;")

dbGetQuery(db, "SELECT COUNT(DISTINCT title)
          FROM spacemovies;")
```

When I looked at the titles, sure enough they sounded space related! To get the years these were released, I used the max and min function in the sql statement. The space movies were released from 1911 to 2018.

```
dbGetQuery(db, "SELECT MIN(production_year), MAX(production_year)
          FROM spacemovies;") # 1911 to 2018
```

I had great difficulties trying to get the top 5 actors in each of the space movies. Then, according to a piazza post, the top 5 refers the the nr_order (or the billing position) of the movie in the cast_info table. To solve this, I combined cast_info and spacemovies and name so I could also get the nr_order column. Then, I used a WHERE statement as a condition so nr_order is between 1 and 5. I know instead of making so many temporary tables, I could just put everything into one dbGetQuery statement, however I have found it is easier for me to do things step by step and it also runs faster when I break it down.

```
nr_order              name                     title
1        1     Franchi, Franco         002 operazione Luna
2        2    Ingrassia, Ciccio         002 operazione Luna
3        3     Randall, Mónica         002 operazione Luna
4        4         Sini, Linda         002 operazione Luna
5        5        Silva, María         002 operazione Luna
6        1      Bodeen, DeWitt              12 to the Moon
7        2         Kobi, Michi              12 to the Moon
8        3         Conway, Tom              12 to the Moon
9        4     Dexter, Anthony              12 to the Moon
10       5       Wengraf, John              12 to the Moon
11       1    Knight, Charlotte    20 Million Miles to Earth
12       2        Taylor, Joan    20 Million Miles to Earth
13       3       Puglia, Frank    20 Million Miles to Earth
14       4       Zaremba, John    20 Million Miles to Earth
15       5 Henry, Thomas Browne    20 Million Miles to Earth
```

```
dbGetQuery(db, "CREATE TEMPORARY TABLE spacecast AS
         SELECT * FROM cast_info, spacemovies
         WHERE cast_info.movie_id = spacemovies.id")
dbGetQuery(db, "SELECT * FROM spacecast LIMIT 5;")

dbGetQuery(db, "CREATE TABLE spacenames AS
         SELECT * FROM spacecast, name
         WHERE spacecast.person_id = name.id")
dbGetQuery(db, "SELECT * FROM spacenames LIMIT 5;")

dbGetQuery(db, "SELECT nr_order, name, title
          FROM spacenames
         WHERE nr_order BETWEEN 1 AND 5
         GROUP BY title, nr_order LIMIT 30")
```

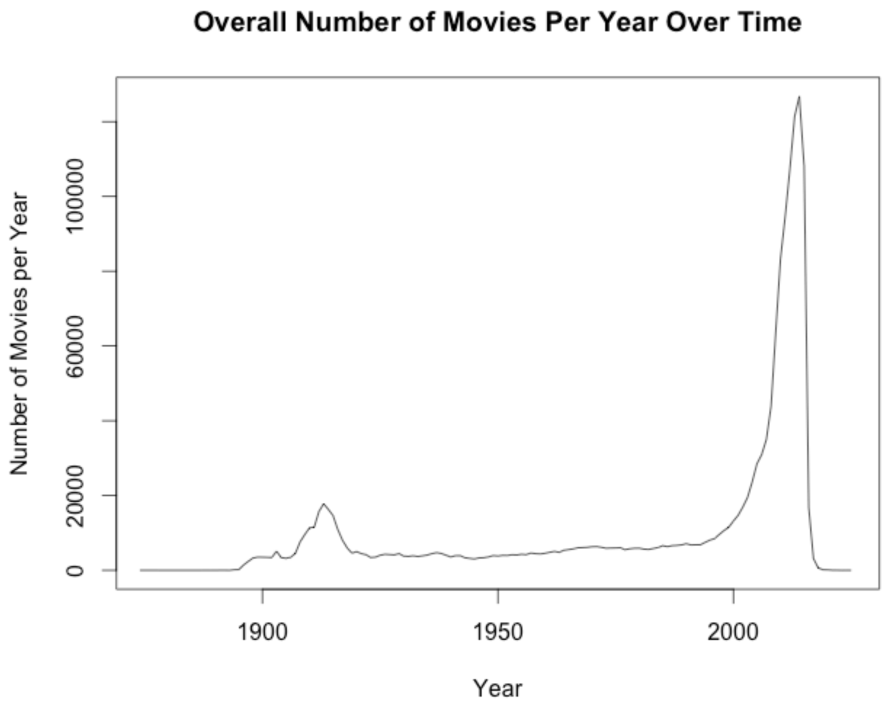
8. **Q8: Has the number of movies in each genre changed over time? Plot the overall number of movies in each year over time, and for each genre.**

Yes the number of movies in each genre has changed over time. It has generally increased. Might be hard to tell in this bit of output, but there does seem to be an increasing trend for each genre at least by looking at just the numbers. In general, as year increases so does the overall number of movies. We can view a plot for specifics.

```
        production_year COUNT(info)       info
      1              NA      24473      Drama
      2            1874          2      Short
      3            1878          1      Short
      4            1887          1      Short
      5            1888          8      Short
      6            1889          3      Short
      7            1890          8      Short
      8            1891         14      Short
```

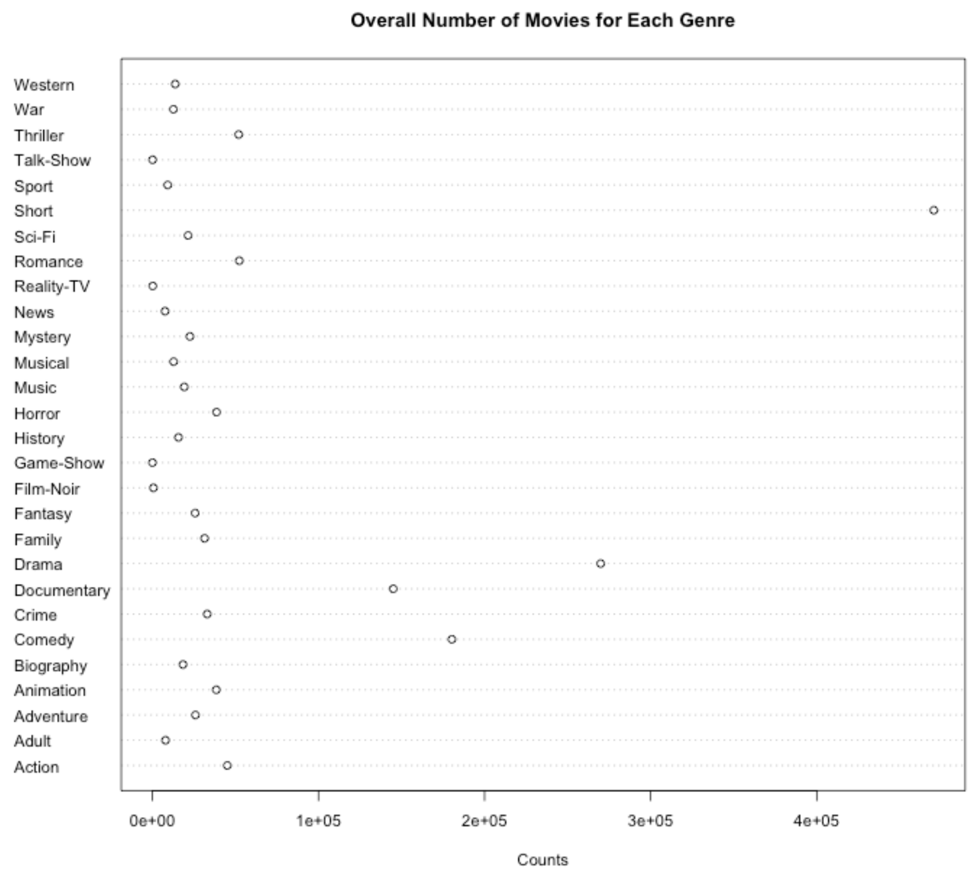
```
9         1892       19     Sport
10         1893        3     Short
11         1894       142     Sport
12         1895       195     Short
13         1896      1400      Short
14         1897      2382      Short
15         1898      3269      Short
16         1899      3523      Short
17         1900      3500        War
18         1901      3460      Short
19         1902      3418      Short
20         1903      5107      Short
```

Here is a plot of the overall number of movies per year over time. We can see there is a very gradual increase up till the 2000s where there is a huge spike in number of movies. The reason for the decrease is probably due to not knowing the true number of movies in the future that hasn't come out yet. Also about 1910s theres a smaller spike which is interesting.



```
movieyr = dbGetQuery(db, "SELECT production_year, COUNT(info)
        FROM genres, title
        WHERE genres.movie_id = title.id
        AND title.kind_id = 1
        GROUP BY production_year;")

colnames(movieyr) = c("Year", "Number")

movieyro = na.omit(movieyr) # remove NA for graphing
plot(movieyro,  main = "Overall Number of Movies Per Year Over Time", xlab = "Year", ylab = "Number of Movies per Year", xlim = c(1875, 2025), type = "l" )
```

Here is a plot of the overall number of movies in each genre

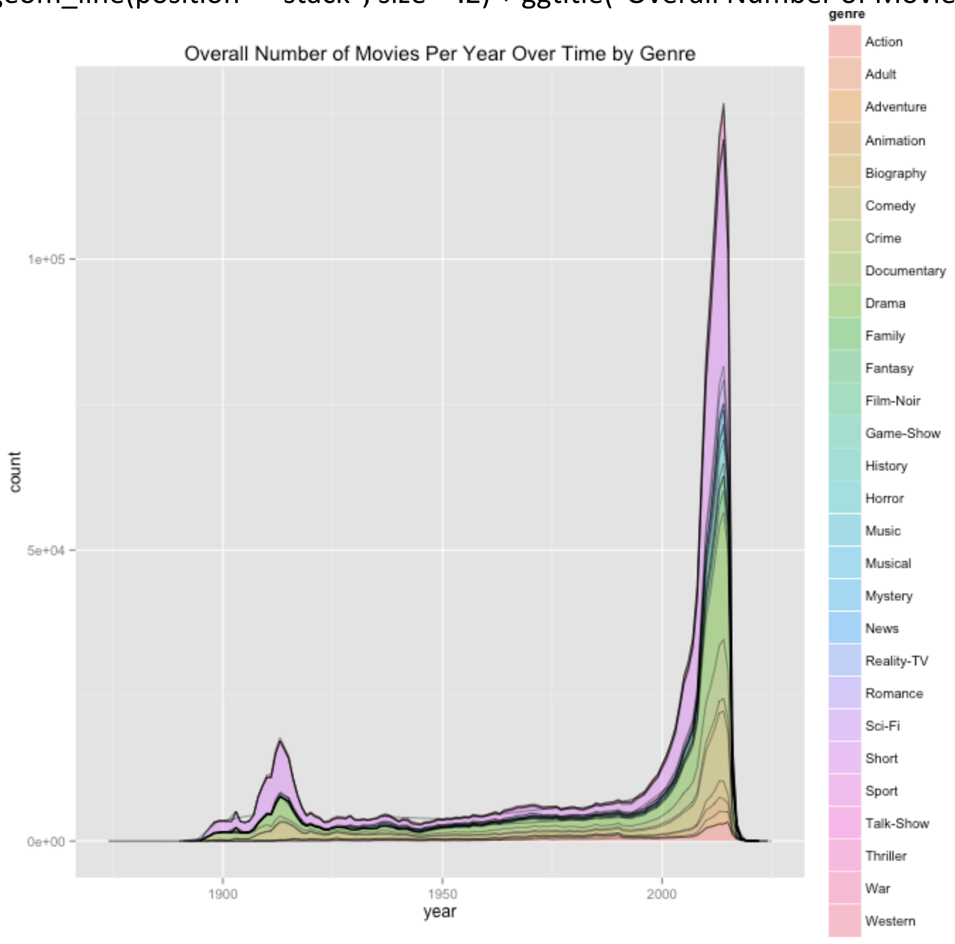


```
genre = dbGetQuery(db, "SELECT info, COUNT(*)
          FROM genres, title
         WHERE genres.movie_id = title.id
         AND title.kind_id = 1
         GROUP BY info ;")
colnames(genre) = c("type", "count")
dotchart(genre$count, labels = as.factor(genre$type), cex = .7, main = "Overall Number of Movies for Each
Genre", xlab = "Counts")
```

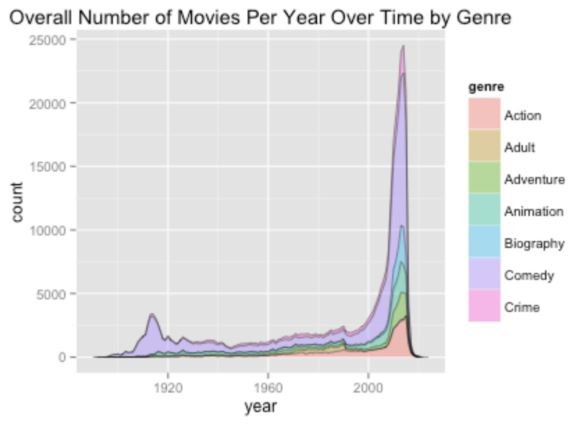
plot of all genres over time

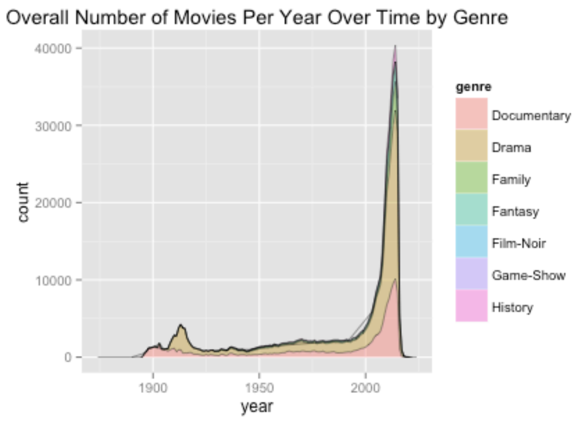
```
genreyr = dbGetQuery(db, "SELECT production_year, info, COUNT(*)
          FROM genres, title
         WHERE genres.movie_id = title.id
         AND title.kind_id = 1
         GROUP BY production_year, info ;")

genreyr = na.omit(genreyr)
colnames(genreyr) = c("year", "genre", "count")

plot(genreyr$year, genreyr$count, main = "Number of Movies per Yr for each Genre", xlab = "Year", ylab =
"Number of Movies", type = "l")

library(ggplot2)
```

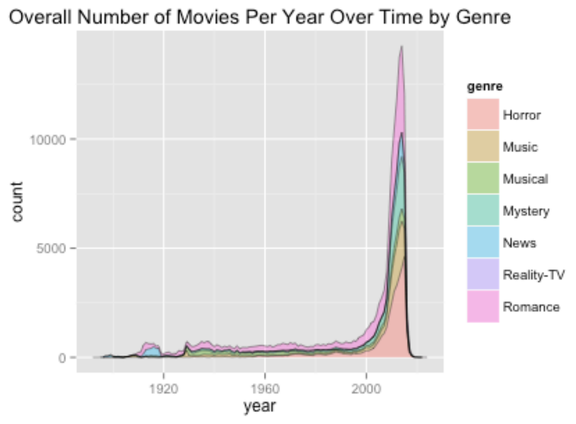
```
ggplot(genreyr, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")
```

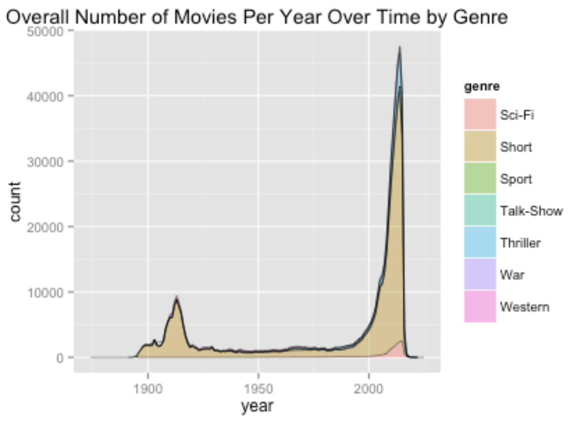


Since the plot is cramped, I split the graph into 4 separate graphs with each one showing counts of 7 genres each. This way, one can easily see how the genres have trended over time compared with certain genres.









```
library(gridExtra)
one = subset(genreyr, genre %in% c("Action", "Adult","Adventure", "Animation", "Biography", "Comedy",
"Crime"))
plot1 = ggplot(one, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")
```

9. **Q9: Who are the actors that have been in the most movies? List the top 20.**

The top 20 actors that have been in the most movies are:

```
               name count(name)
1         Blanc, Mel        2904
2       Brahmanandam         995
3  Onoe, Matsunosuke         930
4       Mercer, Jack         792
5       Hack, Herman         689
6         Harris, Sam        676
7         Phelps, Lee        651
8        Jeremy, Ron         647
9     Kapoor, Shakti         646
10      Cobb, Edmund         641
11   O'Connor, Frank         632
12     Miller, Harold        619
13       London, Tom         612
14       Mower, Jack         599
15  Farnum, Franklyn         596
16      Osborne, Bud         594
```

```
17  Richardson, Jack        586
18     Garcia, Eddie        579
19      Ellis, Frank        574
20      Bhasi, Adoor        566
```
This took a really long time to run and was very frustrating because I wasn't even sure if my SQL statement was correct. Luckily after a Google search, Mel Blanc does seem to appear in the most movies because he was a voice actor. To get this, I merged three tables: cast_info, name, and title by id and filtered out the movies and actors by setting the ids = 1. Then, I group by name and order by count of the name in descending order and got the top 20 by using LIMIT 20.

```
dbGetQuery(db, "SELECT name, person_id, count(name)
        FROM cast_info, name, title
        WHERE cast_info.role_id = 1
         AND name.id = cast_info.person_id
         AND title.kind_id = 1
         AND cast_info.movie_id = title.id
         GROUP BY name
         ORDER BY COUNT(name) DESC LIMIT 20;")
```

When I answered this question in R, I used the smaller dataset titled lean_imdbpy_2010_idx.db. First, I used dbReadTable for each name2, cast_info2, title2 tables similar like what I did in SQL. Then, to get just the actors and movies, I subsetted the data so the kind_id and type_id is 1. I then merged the data sets by the id, but I had to rename the columns so that they were called the same thing. Then, I used aggregate which is similar to sql's group by statement. I sorted the names and got the top 20. These are obviously different than the above since I used the even smaller data set. It seems to be much easier to do everything in one sql command!

```
[1] "Mualimm-Ak, Five"    "Rowen, Thomas"       "Rowen, Thomas"       "Rowen, Thomas"
 [5] "Rowen, Thomas"       "Rowen, Thomas"       "Rowen, Thomas"       "Rowen, Thomas"
 [9] "Rowen, Thomas"       "Rowen, Thomas"       "Sbaraglia, Leonardo" "Sbaraglia, Leonardo"
[13] "Sbaraglia, Leonardo" "Sbaraglia, Leonardo" "Sbaraglia, Leonardo" "Sbaraglia, Leonardo"
[17] "Sbaraglia, Leonardo" "Sbaraglia, Leonardo" "Sbaraglia, Leonardo" "Sbaraglia, Leonardo"
```

10. **Q10: Who are the actors that have had the most number of movies with "top billing", i.e., billed as 1, 2 or 3? For each actor, also show the years these movies spanned?**

Actors that have had the most number of movies with "top billing" with the years the movies spanned.

```
              name COUNT(name) MAX(production_year) MIN(production_year)
1         Blanc, Mel        1551                 2011                 1944
2      Shin, Sung-il         394                 1992                 1960
3 Kerrigan, J. Warren        374                 1934                 1910
4         Moran, Lee         369                 2013                 1912
5        Lyons, Eddie        355                 1924                 1911
```

I merged three tables: cast_info, name, and title. I just realized it is the same table as the previous problem except I wanted top billing so I added a WHERE nr_order is between 1 and 3. Since I also want the range of years the movies spanned, I also added a MIN and MAX of the production_year. The result looks pretty reasonable. Notice how Mel Blanc is the first one, just like in the previous problem where we looked at the actors that have been in the most movies. I think there should be some association between the two.

```
dbGetQuery(db, "SELECT name, COUNT(name), MAX(production_year), MIN(production_year)
         FROM cast_info, name, title WHERE nr_order
         BETWEEN 1 AND 3 AND cast_info.role_id = 1
```

```
        AND name.id = cast_info.person_id
        AND title.id = cast_info.movie_id
       AND title.kind_id = 1
        GROUP BY name
        ORDER BY COUNT(name) DESC LIMIT 5")
```

R output is using the smaller dataset titled lean_imdbpy_2010_idx.db

```
"Knighton, Zachary" "Knighton, Zachary" "Knighton, Zachary" "Knighton, Zachary" "Knighton, Zachary"
2012 2014 2016 2014 2011 2015
```

I basically followed the same procedure as what I did in 9, but I added the additional condition in my subset of 9's data as nr_order <= 3 & nr_order >= 1 so we get the equivalent WHERE statement as the sql.

11. **Q11: Who are the 10 actors that performed in the most movies within any given year? What are their names, the year they starred in these movies and the names of the movies?**

The 10 actors that performed in the most movies within any given year are:

```
production_year                title            name COUNT(production_year)
1          1941          Woody Woodpecker        Blanc, Mel                  164
2          1909            With Her Card     Sennett, Mack                 160
3          1939              Wise Quacks        Blanc, Mel                 145
4          1940 You Ought to Be in Pictures        Blanc, Mel                  144
5          1909            With Her Card Johnson, Arthur V.                 138
6          1942      Who's Who in the Zoo        Blanc, Mel                  131
7          1943       Yankee Doodle Daffy        Blanc, Mel                  121
8          1949            Wise Quackers        Blanc, Mel                 120
9          1909            With Her Card       Moore, Owen                  118
10          1948     You Were Never Duckier        Blanc, Mel                  117
```

The idea for the problem is to find the number movies each actor appeared in for each year and then find the 10 largest counts. So for 2015, maybe one actor appeared in 20 movies. In 2013, another actor appeared in 25 movies. So we need to get the 10 largest of these counts. (from piazza). To do this in sql commands, I first created a table of all the year and actor name pairs with a group by and also wanted the name, year, and count as output. Then, I sorted the count in descending order and the top ten were the actors with most amount of performances in movies in any year of all time. I thought I was supposed to find for every year, the top 10 actors. This way is more tedious, though I did try it (can check the code appendix if interested).

```
dbGetQuery(db, "CREATE TABLE yearactor AS
        SELECT production_year, name, COUNT(*) AS number_of_movies
        FROM title, name, cast_info
        WHERE cast_info.role_id = 1
        AND name.id = cast_info.person_id
        AND title.kind_id = 1
        AND cast_info.movie_id = title.id
        GROUP BY name, production_year")

dbGetQuery(db, "SELECT production_year, name
        FROM yearactor
        ORDER BY number_of_movies DESC LIMIT 10")
```

The name of the movies (some of the output):

```
              title year  actorname
```

```
1             A Coy Decoy 1941 Blanc, Mel
2             A Coy Decoy 1941 Blanc, Mel
3             A Coy Decoy 1941 Blanc, Mel
4           A Helping Paw 1941 Blanc, Mel
5           A Helping Paw 1941 Blanc, Mel
6  All This and Rabbit Stew 1941 Blanc, Mel
7        Andy Panda's Pop 1941 Blanc, Mel
8       Aviation Vacation 1941 Blanc, Mel
9       Aviation Vacation 1941 Blanc, Mel
10      Aviation Vacation 1941 Blanc, Mel
```

```
dbGetQuery(db, "CREATE TEMPORARY TABLE toptens
        AS SELECT production_year AS year, name AS actorname
        FROM yearactor
        ORDER BY number_of_movies DESC LIMIT 10")

# get the names
dbGetQuery(db, "SELECT title, year, actorname
        FROM toptens, nametitlecast
        WHERE toptens.year = nametitlecast.production_year
        AND toptens.actorname = nametitlecast.name LIMIT 10")
```

12**. Q12: Who are the 10 actors that have the most aliases (i.e., see the aka_names table).**
The 10 actors with the most aliases:

```
                              name person_id COUNT(person_id)
1                  Manera, Jesús Franco    662453             78
2                  Massaccesi, Aristide    444281             71
3                      Bernell, Ursula   2543347             63
4                    Cowen, Harvey Joel   1796694             53
5                        Ho, Chi Kueng    882821             50
6                     Nassivera, Joseph   1869225             42
7  me, I'm on a roll' Albert Kimson 'Butter   4318812             40
8                 Moreno, Nathanael León   1176039             39
9                       Grosso, Gilbert    373754             38
10                      Presová, Zuzana   3154545             38
```

The problem used just one table the aka_names table and was a matter of using GROUP BY and ORDER BY correctly. I grouped by the person_id and ordered by the count of the person_id to get the results.

```
dbGetQuery(db, "SELECT name, person_id, COUNT(person_id)
        FROM aka_name GROUP BY person_id
        ORDER BY COUNT(*) DESC LIMIT 10")
```

Using R with the smaller dataset titled lean_imdbpy_2010_idx.db , the ten actors with most aliases is:

```
name person_id
6293    Alex 322298063
58019  Chris 271398429
217818  Mike 252906404
164760   Kat 244560857
282970   Sam 224447921
```

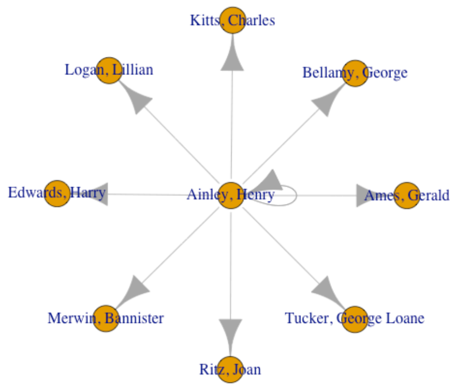
```
153486   Jay 221422916
191482   Liz 187502605
164932 Katie 183103246
234506  Nick 176599685
154211   Jen 176227125
```

I did this by similar procedure as question 9 but I aggregated over person_id and name. this may not be completely correct though because im not sure if aggregate can work for categorical variables, and since we can't use plyr package, this is the best I can do.

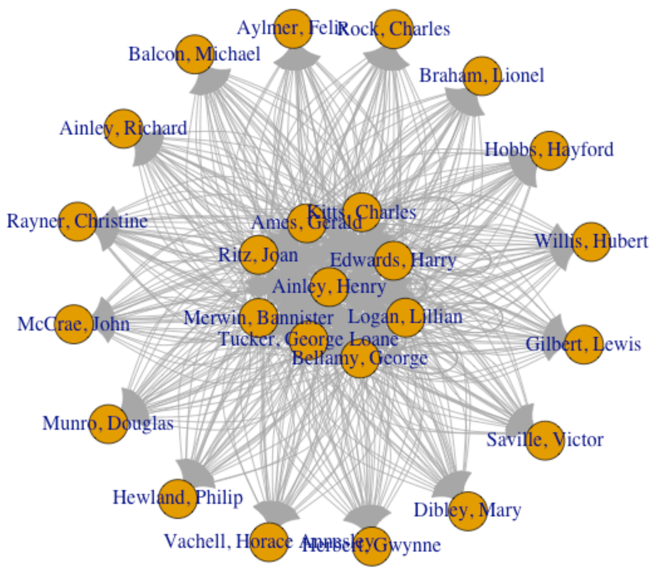
13. **Q13: Networks: Pick a (lead) actor who has been in at least 20 movies. Find all of the other actors that have appeared in a movie with that person. For each of these, find all the people they have appeared in a move with it. Use this to create a network/graph of who has appeared with who. Use the igraph or statnet packages to display this network.**
To pick a lead actor that has been in at least 20 movies, I did the same as question 10 but I looked at nr_order only equal to 1 and made sure the count to be equal to 20 so I don't have to deal with a super big network in the graph later on. I picked the first guy, Henry Ainley, some actor from hundreds of years ago. Then, I pretty much followed Nick's codes from office hours. The methodology is to first get the specific id for Henry. Then, pull all the movies for Henry. Then, get the whole cast which is the actors that were in those movies. Finally, repeat again and this is the 2nd generation. To do this, basically started off with just one id. Then, notice that we can put this in a function with sprintf so dbGetQuery gets called for each id (whether actor or movie). Then, it's a matter of using lapply and messing around with the class of the variables then its ready to be put into igraph. I had a terrible time with this. I felt like there weren't enough examples of this in class, but nevertheless I tried to do it using the nodes and edges method since it seemed easier than the matrix method. It took me hours to finally figure out how to deal with the unequal lengths of the generations and how to use it in igraph. I finally posted on piazza and Duncan helped me clarify it. Basically I need to repeat each element in the previous generation to match the next generation.

```
               name COUNT(name)
1      Ainley, Henry         20
2    Albertson, Frank         20
3      Allan, Richard         20
4       Angelo, Jean         20
5            Ankito         20
6   Bautista, Herbert         20
7         Benny, Jack         20
8    Blanche, Francis         20
9  Bowers, Charles R.         20
10    Buchholz, Horst         20
```

Here is a graph network of my lead actor Henry with the 1st generation (a subset, just the first 10 since my R would keep crashing and fail to even load anything).



Here is a network graph of the 1st generation with the 2nd generation (once again, a subset the first 40). I'm not quite sure why my graph looks so circular. Maybe it is entirely possible some of Henry's actor mates did not act with each other. I mean Henry is from another century anyways. I could also be making the graph wrong, in which case this sucks I spent so much time on this homework!



14. **Q14: What are the 10 television series that have the most number of movie stars appearing in the shows?**

The 10 tv series (which means kind_type is 2 instead of 1). I defined "movie stars" as actors who have nr_order equal to 1 or top billing. Then, I did similar procedure as question 12. Seems like a lot of Spanish tv series have many stars appearing in it. I'm not too surprised since many of these series are super long and have a huge cast and extended families.

```
                              title COUNT(title) COUNT(*)
1            Tout le monde il est gentil          48
2  The Bugs Bunny/Looney Tunes Comedy Hour          42
```

```
3                        Typisch Chris          32
4                           Code MENT          30
5                       Turno da Noite          24
6                          Ved pejsen          22
7                  ¿Es usted el asesino?          21
8                     Tonny Toupé show          21
9                               Zorro          18
10                         TV akademiet          18
```

```
dbGetQuery(db, "SELECT title, COUNT(*)
         FROM cast_info, name, title
          WHERE nr_order = 1
         AND cast_info.role_id = 1
         AND name.id = cast_info.person_id
         AND title.id = cast_info.movie_id
         AND title.kind_id = 2
         GROUP BY name
         ORDER BY COUNT(*) DESC LIMIT 10")
```

Conclusion

SQL commands are pretty simple and few and can be used to extract large amount of data efficiently. SQL commands take maybe one long line whereas the same question needs to be answered in many steps using just R. I definitely learned a lot more about SQL but this assignment was a headache. There were so many data sets uploaded at different times. I had already worked on a few problems from the original data set, and had to redo work.

Code Appendix

```
# Assignment 5
# tiffany chen

library(RSQLite)
## old database
db = dbConnect(drv = SQLite(), dbname = '/Users/tiffanychen/Desktop/STA 141/imdb_data')
dbListTables(db)

## Q1
# look specifically at actors
dbListFields(db, "actors")

# count how many actors 3500167
dbGetQuery(db, "SELECT count(idactors) FROM actors")
dbGetQuery(db, "SELECT count(fname) FROM actors") # 3500166
dbGetQuery(db, "SELECT idactors FROM actors")

# number of movies 1298737
dbListFields(db, "movies")
dbGetQuery(db, "SELECT count(title) FROM movies")
dbGetQuery(db, "SELECT count(idmovies) FROM movies") # 1,298,737
```

```
## Q2
dbListFields(db, "movies")
dbGetQuery(db, "SELECT year FROM movies")
dbGetQuery(db, "SELECT MIN(year) FROM movies") # 1
dbGetQuery(db, "SELECT MAX(year) FROM movies") # 2025

dbGetQuery(db, "SELECT DISTINCT(year) FROM movies") # lists all the distinct

year_period = unique(dbGetQuery(db, "SELECT year FROM movies"))
sort(year_period$year)

dbGetQuery(db, "SELECT year, title FROM movies LIMIT 100")

dbGetQuery(db, "SELECT * FROM movies WHERE year = '2025'") # makes sense
dbGetQuery(db, "SELECT * FROM movies WHERE year = '1'")
dbGetQuery(db, "SELECT * FROM movies WHERE year = '3'")

## Q3
dbGetQuery(db, "SELECT DISTINCT gender FROM actors")

# ============================================================
## new data
db = dbConnect(drv = SQLite(), dbname = '/Users/tiffanychen/Desktop/STA 141/lean_imdbpy.db')
dbListTables(db)

## data that is even smaller (used for q9-12 for using R only)
db = dbConnect(drv = SQLite(), dbname = '/Users/tiffanychen/Desktop/STA 141/lean_imdbpy_2010_idx.db')

## Q1
# How many actors are there in the database? How many movies?
dbGetQuery(db, "SELECT COUNT(DISTINCT person_id)
        FROM aka_name;") # 719127 distinct names of actors

dbGetQuery(db, "SELECT COUNT(name)
        FROM name LIMIT 10;") # 5375509 actors

dbGetQuery(db, "SELECT COUNT(*)
        FROM title LIMIT 10;") # 3527732 movies/titles

dbGetQuery(db, "SELECT COUNT(DISTINCT(title))
        FROM title
        WHERE kind_id = 1;") # 878800 movies

dbGetQuery(db, "SELECT COUNT( DISTINCT(name))
       FROM cast_info, name
       WHERE role_id = 1
        AND name.id = cast_info.person_id;") # 1936807 actors

## Q2
```

```
# What time period does the database cover?
# not just movies
dbGetQuery(db, "SELECT MIN(production_year)
        FROM title;") # 1874 min yr
dbGetQuery(db, "SELECT MAX(production_year)
        FROM title;") # 2025 max yr

dbGetQuery(db, "SELECT * FROM title
        WHERE production_year = 1874") # looking at it to check realness
dbGetQuery(db, "SELECT title FROM title
        WHERE production_year = 2018")

## Q3
# What proportion of the actors are female? male?
dbGetQuery(db,"SELECT * FROM name LIMIT 10;")

# returns the numbers of males/females/NAs
dbGetQuery(db, "SELECT gender,
         COUNT(*) * 100.0 / (SELECT COUNT(*) FROM name)
        FROM name GROUP BY gender;") # actual proportions

## Q4
# merge kind of movies with the title table
dbGetQuery(db, "CREATE TABLE movie AS SELECT *
      FROM title AS t, kind_type AS k
        WHERE t.kind_id = k.id;")
dbGetQuery(db, "SELECT * FROM movie LIMIT 10")

# return proportions of entries
dbGetQuery(db, "SELECT kind, COUNT(*) * 100.0 /
         (SELECT COUNT(*) FROM movie)
        FROM movie GROUP BY kind;")

## Q5
dbGetQuery(db, "SELECT * FROM info_type LIMIT 10;") # genres = 3
dbGetQuery(db, "SELECT * FROM movie_info LIMIT 10;")

# only look at genres which is info_type 3
dbGetQuery(db, "SELECT DISTINCT info
        FROM movie_info
        WHERE info_type_id = 3;")

## Q6
# http://stackoverflow.com/questions/12235595/find-most-frequent-value-in-sql-column
dbGetQuery(db, "SELECT info, COUNT(info)
         FROM movie_info
         WHERE info_type_id = 3
        GROUP BY info
        ORDER BY COUNT(*) DESC LIMIT 10;")
```

```
## Q7
dbGetQuery(db, "CREATE TABLE keywordtitle AS
        SELECT * FROM title, movie_keyword, keyword
        WHERE title.id = movie_keyword.movie_id
        AND movie_keyword.keyword_id = keyword.id")

dbGetQuery(db, "SELECT * FROM keywordtitle LIMIT 5;")

dbGetQuery(db, "CREATE TABLE spacemovies AS
          SELECT * FROM keywordtitle
          WHERE keyword = 'space' AND kind_id = 1;")

dbGetQuery(db, "SELECT COUNT(DISTINCT title)
         FROM spacemovies;") # 400 space movies
dbGetQuery(db, "SELECT * FROM spacemovies LIMIT 5")
# ===========================================================================
# use production_year from title table
dbGetQuery(db, "SELECT production_year FROM spacemovies;")
dbGetQuery(db, "SELECT MIN(production_year), MAX(production_year)
         FROM spacemovies;") # 1911 to 2018
# ==================================================

dbGetQuery(db, "CREATE TEMPORARY TABLE spacecast AS
         SELECT * FROM cast_info, spacemovies
         WHERE cast_info.movie_id = spacemovies.id")
dbGetQuery(db, "SELECT * FROM spacecast LIMIT 5;")

dbGetQuery(db, "CREATE TABLE spacenames AS
         SELECT * FROM spacecast, name
         WHERE spacecast.person_id = name.id")
dbGetQuery(db, "SELECT * FROM spacenames LIMIT 5;")

dbGetQuery(db, "SELECT nr_order, name, title
          FROM spacenames
         WHERE nr_order BETWEEN 1 AND 5
         GROUP BY title, nr_order LIMIT 30")

dbGetQuery(db,"CREATE TABLE spaceactors AS
         SELECT * FROM spacenames
         WHERE role_id = 1;")

dbGetQuery(db, "SELECT nr_order, name, title
         FROM spacenames LIMIT 50")

# ==================================================
## Q8
movieyr = dbGetQuery(db, "SELECT production_year, COUNT(info)
        FROM genres, title
        WHERE genres.movie_id = title.id
        AND title.kind_id = 1
        GROUP BY production_year;")

# Plot the overall number of movies in each year over time,
head(movieyr)
colnames(movieyr) = c("Year", "Number")

movieyro = na.omit(movieyr) # remove NA for graphing
plot(movieyro,  main = "Overall Number of Movies Per Year Over Time", xlab = "Year", ylab = "Number of
Movies per Year", xlim = c(1875, 2025), type = "l" )

# same but using ggplot, connect lines instead of dots
library(ggplot2)
ggplot(movieyro, aes(Year, Number)) + geom_line()  + xlab("Year") + ylab("Number of Movies per Year") +
ggtitle("Overall Number of Movies Per Year Over Time")

# http://stackoverflow.com/questions/2421388/using-group-by-on-multiple-columns
# returns year and count of movies by genre
genre = dbGetQuery(db, "SELECT info, COUNT(*)
         FROM genres, title
        WHERE genres.movie_id = title.id
        AND title.kind_id = 1
        GROUP BY info ;")
colnames(genre) = c("type", "count")
dotchart(genre$count, labels = as.factor(genre$type), cex = .7, main = "Overall Number of Movies for Each
Genre", xlab = "Counts")

# plot over time, per genre
genreyr = dbGetQuery(db, "SELECT production_year, info, COUNT(*)
         FROM genres, title
        WHERE genres.movie_id = title.id
        AND title.kind_id = 1
        GROUP BY production_year, info ;")

head(genreyr, 40)
genreyr = na.omit(genreyr)
colnames(genreyr) = c("year", "genre", "count")

plot(genreyr$year, genreyr$count, main = "Number of Movies per Yr for each Genre", xlab = "Year", ylab =
"Number of Movies", type = "l")

library(ggplot2)
ggplot(genreyr, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")

## I WANT TO DO 4 GRAPHS, each graph has 7 genres
library(gridExtra)
```

```
one = subset(genreyr, genre %in% c("Action", "Adult","Adventure", "Animation", "Biography", "Comedy",
"Crime"))
plot1 = ggplot(one, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")

two = subset(genreyr, genre %in% c("Documentary", "Drama", "Family","Fantasy", "Film-Noir", "Game-Show",
"History"))
plot2 = ggplot(two, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")

three = subset(genreyr, genre %in% c("Horror", "Music","Musical", "Mystery", "News", "Reality-TV",
"Romance"))
plot3 = ggplot(three, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")

four = subset(genreyr, genre %in% c("Sci-Fi", "Short","Sport", "Talk-Show", "Thriller", "War", "Western"))
plot4 = ggplot(four, aes(x = year, y = count, fill = genre)) + geom_area(colour = NA, alpha = .4) +
geom_line(position = "stack", size = .2) + ggtitle("Overall Number of Movies Per Year Over Time by Genre")

grid.arrange(plot1, plot2, plot3, plot4, ncol = 2, nrow = 2)
# https://www.safaribooksonline.com/library/view/r-graphics-cookbook/9781449363086/ch04.html

## Q9
# actors and names
dbGetQuery(db, "SELECT name, person_id, count(name)
        FROM cast_info, name, title
        WHERE cast_info.role_id = 1
         AND name.id = cast_info.person_id
         AND title.kind_id = 1
         AND cast_info.movie_id = title.id
         GROUP BY name
         ORDER BY COUNT(name) DESC LIMIT 20;")

## USING R
## data that is even smaller
db = dbConnect(drv = SQLite(), dbname = '/Users/tiffanychen/Desktop/STA 141/lean_imdbpy_2010_idx.db')
# http://stackoverflow.com/questions/18799901/data-frame-group-by-column
dbListTables(db)
name = dbReadTable(db, "name2")
head(name)
cast = dbReadTable(db, "cast_info2")
title = dbReadTable(db, "title2")
head(cast)
head(title)

actors = subset(cast, role_id == "1") # actors only
movies = subset(title, kind_id == "1")

colnames(name)[1] = "person_id"
```

```
actorname = merge(actors, name, by = c("person_id"))

colnames(movies)[1] = "movie_id"
actornametitle = merge(actorname, movies, by = c("movie_id"))

sums = aggregate(person_id ~ movie_id, actornametitle, sum) # similar to GROUP BY
attach(sum)
newdata = sum[order(-movie_id), ]
head(newdata, 20)

library(data.table) # results confirmed
dt = data.table(actornametitle)
dt[, sum(person_id), by = movie_id]

## Q10
dbGetQuery(db, "SELECT name, COUNT(name), MAX(production_year), MIN(production_year)
        FROM cast_info, name, title WHERE nr_order
        BETWEEN 1 AND 3 AND cast_info.role_id = 1
        AND name.id = cast_info.person_id
        AND title.id = cast_info.movie_id
       AND title.kind_id = 1
        GROUP BY name
        ORDER BY COUNT(name) DESC LIMIT 5")

dbGetQuery(db, "SELECT * FROM title LIMIT 5")
dbGetQuery(db, "SELECT * FROM cast_info LIMIT 5")
dbGetQuery(db, "SELECT * FROM name LIMIT 5")

## Q11
# for each year
dbGetQuery(db, "CREATE TABLE yearactor AS
        SELECT production_year, name, COUNT(*) AS number_of_movies
        FROM title, name, cast_info
        WHERE cast_info.role_id = 1
        AND name.id = cast_info.person_id
        AND title.kind_id = 1
        AND cast_info.movie_id = title.id
        GROUP BY name, production_year")

dbGetQuery(db, "CREATE TABLE yearactor2 AS
        SELECT production_year, name, number_of_movies,
        (SELECT COUNT(*)
        FROM yearactor
        WHERE production_year = t1.production_year
        AND number_of_movies >= t1.number_of_movies) AS rank
        FROM yearactor AS t1")

dbGetQuery(db, "SELECT production_year, name, number_of_movies
        FROM yearactor2
        WHERE rank <= 10")

# for all time
dbGetQuery(db, "CREATE TABLE yearactor AS
        SELECT production_year, name, COUNT(*) AS number_of_movies
        FROM title, name, cast_info
        WHERE cast_info.role_id = 1
        AND name.id = cast_info.person_id
        AND title.kind_id = 1
        AND cast_info.movie_id = title.id
        GROUP BY name, production_year")

dbGetQuery(db, "SELECT production_year, name
        FROM yearactor
        ORDER BY number_of_movies DESC LIMIT 10")

# specific movies
dbGetQuery(db, "CREATE TEMPORARY TABLE toptens
        AS SELECT production_year AS year, name AS actorname
        FROM yearactor
        ORDER BY number_of_movies DESC LIMIT 10")

# get the names
dbGetQuery(db, "SELECT title, year, actorname
        FROM toptens, nametitlecast
        WHERE toptens.year = nametitlecast.production_year
        AND toptens.actorname = nametitlecast.name LIMIT 10")

# i just realized Q9-11 uses the same 3 tables
# title, name, and cast_info....
# i'll just create a table
dbGetQuery(db, "CREATE TABLE nametitlecast AS SELECT *
        FROM title, name, cast_info
        WHERE cast_info.role_id = 1
        AND name.id = cast_info.person_id
        AND title.kind_id = 1
        AND cast_info.movie_id = title.id ")

dbGetQuery(db, "SELECT production_year, title, name, COUNT(DISTINCT title)
        FROM nametitlecast
        GROUP BY production_year, name
        ORDER BY COUNT(DISTINCT title) DESC LIMIT 10")

dbGetQuery(db, "SELECT production_year, title, name, COUNT(production_year)
        FROM nametitlecast
        GROUP BY production_year, name
        ORDER BY COUNT(production_year) DESC LIMIT 10")
```

```
dbGetQuery(db, "SELECT * FROM nametitlecast LIMIT 5")
dbGetQuery(db, "SELECT * FROM name LIMIT 5")
dbGetQuery(db, "SELECT * FROM cast_info LIMIT 5")

# USING R
actornametitle = merge(actorname, movies, by = c("movie_id"))
# http://stackoverflow.com/questions/27193373/what-is-the-r-equivalent-of-sql-select-from-table-group-by-
c1-c2
sums = aggregate(. ~ production_year + name, actornametitle, FUN = head, 1)
attach(sum)
newdata = sum[order(-movie_id), ]
head(newdata, 20)

## Q12
# Who are the 10 actors that have the most aliases (i.e., see the aka_name table).
dbGetQuery(db, "SELECT * FROM aka_name LIMIT 10;")

dbGetQuery(db, "SELECT name, person_id, COUNT(person_id)
        FROM aka_name GROUP BY person_id
        ORDER BY COUNT(person_id) DESC LIMIT 10")

dbGetQuery(db, "SELECT name, person_id, COUNT(person_id)
        FROM aka_name GROUP BY person_id
        ORDER BY COUNT(*) DESC LIMIT 10")

## Q13
dbListTables(db)
# top lead actor been in 20 movies.
dbGetQuery(db, "SELECT name, COUNT(name)
        FROM cast_info, name, title
         WHERE nr_order = 1
         AND cast_info.role_id = 1
        AND name.id = cast_info.person_id
        AND title.id = cast_info.movie_id
       AND title.kind_id = 1
        GROUP BY name
        HAVING COUNT(*) = 20 LIMIT 10")

# Ainley, Henry first person
# these steps are from nick's OH
# 1. pull person_id for specific actor
ids = dbGetQuery(db, "SELECT id FROM name
            WHERE name = 'Ainley, Henry'")
ids

# 2. pull all movies for that actor
henrymovies = dbGetQuery(db, "SELECT movie_id
                FROM cast_info
                WHERE person_id = 21906")
```

```
# function
find_movies = function(actor_id, db) {
  # get movies for a specific person ID
  qr = sprintf('SELECT movie_id FROM cast_info
             WHERE person_id = %i', actor_id)
  dbGetQuery(db, qr)
}

henrymovies = find_movies(ids$id, db)


# 3. pull cast for all that actor's movies
qr = sprintf('SELECT name FROM cast_info, name
             WHERE cast_info.movie_id = %i
             AND name.id = cast_info.person_id', henrymovies$movie_id)
dbGetQuery(db, qr)

# function
find_actors = function(x, db) {
  # find all actors for a given movie
  qr = sprintf('SELECT name FROM cast_info, name
             WHERE cast_info.movie_id = %i
             AND name.id = cast_info.person_id', x)
  dbGetQuery(db, qr)
}

find_actorsid = function(x, db) {
  # find all actors for a given movie
  qr = sprintf('SELECT person_id FROM cast_info, name
             WHERE cast_info.movie_id = %i
             AND name.id = cast_info.person_id', x)
  dbGetQuery(db, qr)
}

actors = find_actors(henrymovies$movie_id, db)
actors = find_actorsid(henrymovies$movie_id, db)

movies = unique(henrymovies$movie_id)
cast = lapply(movies, find_actors, db)
cast = lapply(movies, find_actorsid, db)


# 4. repeat (so write a function for 1-3)
# names
cast1 = unlist(cast, use.names = FALSE)
ids2 = lapply(cast1, find_movies, db)

# ids
cast1 = unlist(cast, use.names = FALSE)
```

```
ids = lapply(cast1, find_movies, db)

# 2nd gen
ids2 = unlist(ids, use.names = FALSE)
cast2 = lapply(ids2, find_actors, db) # actors of actors of henry
cast3 = unlist(cast2, use.names = FALSE)
# graphing
library(igraph)

henry = "Ainley, Henry"
cast = cast1[1:10]
e = data.frame(henry = rep(henry, 10), cast)
g = graph_from_data_frame(e, directed = TRUE)
plot(g)

cast3 = cast3[1:30]
gen1 = rep(cast, each = 30)
gen2 = rep(cast3, 30)
e = data.frame(gen1, gen2)
gr = graph_from_data_frame(e, directed = TRUE)
plot(gr)
```