Assignment 5

Huang Fang 913439658

November 30, 2015

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.

The data I use is lean_imdbpy.db, the database with 8 tables dropped

```
library(RSQLite)
library(combinat)
library(igraph)

conn = dbConnect(SQLite(), "/Users/fangh/Downloads/lean_imdbpy.db")
```

To make the computation simpler and faster, I will create a table with title, cast_info, name to select movies and actors.

```
sql = "CREATE TEMPORARY TABLE movie_cast_name AS
       SELECT title.title AS title, kind_id, production_year, person_id, movie_id, role_id,
       nr_order, name FROM title, cast_info, name
       WHERE title.id = cast_info.movie_id AND cast_info.person_id = name.id
       AND kind_id = 1 AND (role_id = 1 OR role_id = 2)"
dbSendQuery(conn, sql)
```

Question 1

How many actors are there in the database? How many movies?

Using function count to select.

```
> q1sql = "SELECT COUNT(DISTINCT person_id) FROM cast_info WHERE role_id = 1 OR role_id = 2"
> num_actors = dbGetQuery(conn, q1sql)
> num_actors
  COUNT(DISTINCT person_id)
1                   3492018

> q1sql = "SELECT COUNT(id) FROM title"
> num_movies = dbGetQuery(conn, q1sql)
> num_movies
  COUNT(id)
1   3527732
```

Question 2

What time period does the database cover?

Using function MAX and MIN to tables which include information about year.

```
> q2sql = "SELECT MIN(production_year) AS MIN, MAX(production_year) AS MAX FROM title"
> year_range = dbGetQuery(conn, q2sql)
> year_range
   MIN  MAX
1 1874 2025
```

However, this is only the year range in the table title, we should explore more. The field ”production_year” in aka_title also has information about year, and the field ”series_years” also has years.

```
> q2sql = "SELECT MIN(production_year) AS MIN, MAX(production_year) AS MAX FROM aka_title"
> year_range = dbGetQuery(conn, q2sql)
> year_range
   MIN  MAX
1 1875 2022

> series_years = dbGetQuery(conn, "SELECT series_years FROM title WHERE series_years <> 'NA'")
> dim(series_years)
[1] 124431      1
> head(series_years)
  series_years
1    1994-1995
2    2006-????
3    2014-????
4    2016-2016
5    2013-????
6    2015-????
```

Using regular expression to get the year range.

```
> pat = "[0-9]{4}"
> all_year = regmatches(series_years[,1],gregexpr(pat, series_years[,1]))
> all_year = unlist(all_year)
> all_year = as.numeric(all_year)
> year_range = c(min(all_year), max(all_year))
> names(year_range) = c("MIN", "MAX")
> year_range
 MIN  MAX
1100 2025
```

So the maximum year is 2025, the minimum year is 1100.

Question 3

What proportion of the actors are female? male?

Using function GOUP BY. Method 1, use SQL and R.

```
> q3sql = "SELECT gender, COUNT(DISTINCT person_id) AS proportion FROM cast_info, name
+ WHERE cast_info.person_id = name.id AND (cast_info.role_id = 1 OR cast_info.role_id = 2)
+ GROUP BY name.gender"
> actor_by_gender  = dbGetQuery(conn, q3sql)
> actor_by_gender$proportion = actor_by_gender$proportion/sum(actor_by_gender$proportion)
> actor_by_gender
  gender proportion
1      f  0.3537021
2      m  0.6462979
```

Method 2, only use SQL

```
> q3sql = "SELECT gender, COUNT(DISTINCT person_id)/(SELECT COUNT(DISTINCT person_id)*1.0
+ FROM cast_info, name WHERE cast_info.person_id = name.id AND
+ (cast_info.role_id = 1 OR cast_info.role_id = 2)) AS proportion FROM cast_info, name
+ WHERE cast_info.person_id = name.id AND (cast_info.role_id = 1 OR cast_info.role_id = 2)
+ GROUP BY name.gender"
> dbGetQuery(conn, q3sql)
  gender proportion
1      f  0.3537021
2      m  0.6462979
```

We get the same result, however, in this way, the SQL is hard to read, and it takes more time on computation.

Question 4

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

Connect title and kind_type to apply function GROUP BY. Method1, use SQL and R

```
> q4sql = "SELECT kind_id, kind, COUNT(DISTINCT title.id) AS proportion FROM title, kind_type
+ where title.kind_id = kind_type.id
+ GROUP BY title.kind_id"
> movie_by_kind = dbGetQuery(conn, q4sql)
> movie_by_kind$proportion = movie_by_kind$proportion/sum(movie_by_kind$proportion)
> movie_by_kind
  kind_id        kind  proportion
1       1       movie 0.249111894
2       2   tv series 0.035273371
3       3    tv movie 0.034126175
4       4 video movie 0.041563815
5       6  video game 0.004341033
6       7     episode 0.635583712
```

Method2, only use SQL

```
> q4sql = "SELECT kind_id, kind, (COUNT(DISTINCT title.id)/(SELECT COUNT(DISTINCT title.id)*1.0
+ FROM title, kind_type WHERE title.kind_id = kind_type.id)) AS proportion
+ FROM title, kind_type
+ WHERE title.kind_id = kind_type.id
+ GROUP BY title.kind_id"
> dbGetQuery(conn, q4sql)
  kind_id        kind  proportion
1       1       movie 0.249111894
2       2   tv series 0.035273371
3       3    tv movie 0.034126175
4       4 video movie 0.041563815
5       6  video game 0.004341033
6       7     episode 0.635583712
```

Of course, we get the same result.

Question 5

How many genres are there? What are their names/descriptions?

Connect table movie_info and info_type and select distinct movie_info.

```
> q5sql = "SELECT DISTINCT movie_info.info FROM movie_info, info_type
+ WHERE movie_info.info_type_id = info_type.id AND  info_type.info = 'genres'"
```

```
> value_genres = dbGetQuery(conn, q5sql)
> nrow(value_genres)
[1] 32
```

There are 32 distinct movie_info.info with info.type = 'genres'

```
> value_genres$info
 [1] "Documentary"  "Reality-TV"   "Horror"
 [4] "Drama"        "Comedy"       "Musical"
 [7] "Talk-Show"    "Mystery"      "News"
[10] "Sport"        "Sci-Fi"       "Romance"
[13] "Family"       "Short"        "Biography"
[16] "Music"        "Game-Show"    "Adventure"
[19] "Crime"        "War"          "Fantasy"
[22] "Thriller"     "Animation"    "Action"
[25] "History"      "Adult"        "Western"
[28] "Lifestyle"    "Film-Noir"    "Experimental"
[31] "Commercial"   "Erotica"
```

If we want to find the distinct movie.info from movies, then we only need to do a little change.

```
> q5sql = "SELECT DISTINCT movie_info.info FROM movie_info, info_type, title
+ WHERE movie_info.info_type_id = info_type.id AND movie_info.movie_id = title.id AND
+ info_type.info = 'genres' AND title.kind_id = 1"
> movie_info = dbGetQuery(conn, q5sql)
> movie_info$info
 [1] "Comedy"      "Short"       "Drama"       "Animation"
 [5] "History"     "War"         "Horror"      "Adventure"
 [9] "Sci-Fi"      "Biography"   "Documentary" "Family"
[13] "News"        "Action"      "Romance"     "Musical"
[17] "Sport"       "Fantasy"     "Mystery"     "Thriller"
[21] "Music"       "Crime"       "Western"     "Adult"
[25] "Film-Noir"   "Reality-TV"  "Game-Show"   "Talk-Show"
```

Now there are 28 distinct movie_info

Question 6

List the 10 most common genres of movies, showing the number of movies in each of these genres.

Only Count those genres with title.kind_id = 1 (only choose from movies)

```
> q6sql = "SELECT movie_info.info, COUNT(movie_info.id) AS number FROM movie_info, title, info_type
+ WHERE movie_info.info_type_id = info_type.id AND  movie_info.movie_id = title.id
+ AND info_type.info = 'genres' AND title.kind_id = 1
+ GROUP BY movie_info.info
+ ORDER BY COUNT(movie_info.id) DESC LIMIT 10"
> dbGetQuery(conn, q6sql)
          info number
1        Short 470488
2        Drama 269898
3       Comedy 180315
4  Documentary 145018
5      Romance  52324
6     Thriller  51961
7       Action  45077
8       Horror  38620
```

```
9    Animation  38461
10       Crime  33010
```

This is our 10 most common genres, the number is the number of movies.

Question 7

Find all movies with the keyword 'space'. How many are there?

Connect movie_keyword, title and keyword, and select distinct movie_id.

```
> q7sql = "SELECT DISTINCT movie_keyword.movie_id FROM movie_keyword, title, keyword
+ WHERE movie_keyword.movie_id = title.id AND movie_keyword.keyword_id = keyword.id
+ AND keyword.keyword = 'space' AND title.kind_id = 1"
> movies_key_space = dbGetQuery(conn, q7sql)
> nrow(movies_key_space)
[1] 401
> head(movies_key_space)
  movie_id
1  2365979
2  2367917
3  2371167
4  2371436
5  2371922
6  2376022
```

There are 401 movies with keyword 'sapce'.

What are the years these were released?

Find their distinct production_year.

```
> q7sql = "SELECT DISTINCT production_year FROM title WHERE id IN
+ (SELECT DISTINCT movie_keyword.movie_id FROM movie_keyword, title, keyword
+ WHERE movie_keyword.movie_id = title.id AND movie_keyword.keyword_id = keyword.id
+ AND keyword.keyword = 'space' AND title.kind_id = 1)
+ ORDER BY production_year ASC"
> movie_year = dbGetQuery(conn, q7sql)
> movie_year$production_year
 [1]   NA 1911 1918 1922 1925 1930 1946 1947 1950 1951 1953
[12] 1954 1955 1956 1957 1958 1959 1960 1961 1962 1964 1965
[23] 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1977
[34] 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
[45] 1989 1990 1991 1992 1993 1994 1996 1997 1998 1999 2000
[56] 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011
[67] 2012 2013 2014 2015 2016 2017 2018
```

Who were the top 5 actors in each of these movies?

Select top 5 actors.

```
> q7sql = "SELECT DISTINCT movie_id, person_id, name, nr_order FROM cast_info, name
+ WHERE cast_info.person_id = name.id AND movie_id IN
+ (SELECT DISTINCT movie_keyword.movie_id FROM movie_keyword, title, keyword
+ WHERE movie_keyword.movie_id = title.id AND movie_keyword.keyword_id = keyword.id
+ AND keyword.keyword = 'space' AND title.kind_id = 1) AND (role_id = 1 OR role_id = 2)
+ AND (nr_order >=1 AND nr_order <=5)
+ ORDER BY movie_id, nr_order"
```

```
> top5 = dbGetQuery(conn, q7sql)
> head(top5)
  movie_id person_id              name nr_order
1  2365979    661113   Franchi, Franco        1
2  2365979    935665 Ingrassia, Ciccio        2
3  2365979   3172528   Randall, Mnica        3
4  2365979   3291555       Sini, Linda        4
5  2365979   3286328      Silva, Mara        5
6  2367917    374630         Clark, Ken        1
> dim(top5)
[1] 1092    4
```

However this method is not perfect, for many movies we may have multiple actors with nr_order = 1, so the number of actors we extract for each movie could be more than 5.
Check if this problem exist.

```
> top5_split = split(top5, top5$movie_id)
> table(sapply(top5_split, length) > 5)

FALSE
  239
```

There is no such problem, however since we cannot guarantee this problem will not happen in another database, so it is not a perfect method.
After checking a lot on google, I find it as a greatest-n-per-group problem.
It is hard to solve directly from SQL. However we can use *lapply* in R to do it, in this way, we have to do the "select" for 401 times, and it will cost us more than 40min. So I will just give the concept without results.

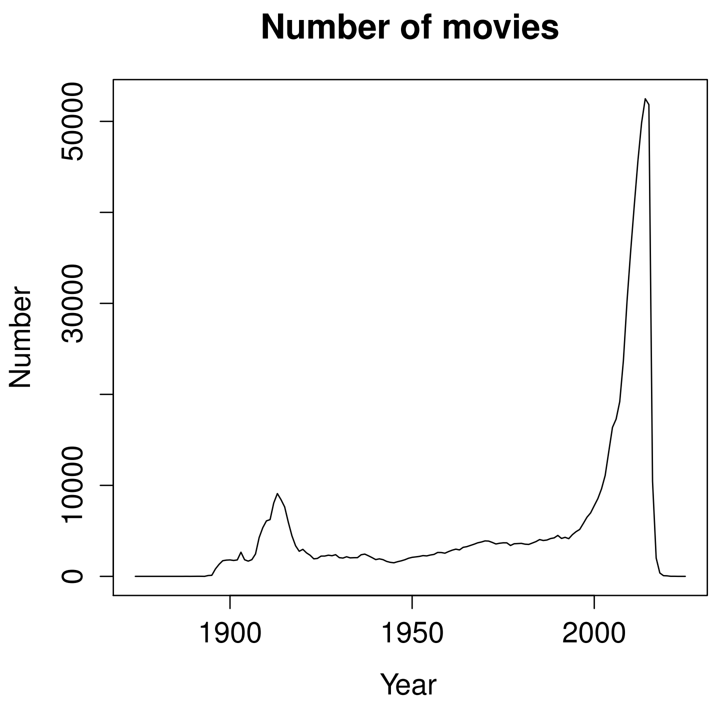
```
movie_list = "SELECT DISTINCT movie_keyword.movie_id FROM movie_keyword, title, keyword
              WHERE movie_keyword.movie_id = title.id AND movie_keyword.keyword_id = keyword.id
              AND keyword.keyword = 'space' AND title.kind_id = 1"
top5_R = lapply(movie_list, function(x)
            dbGetQuery(conn, paste("SELECT DISTINCT movie_id, person_id, name, nr_order
            FROM cast_info, name
            WHERE cast_info.person_id = name.id AND movie_id = ", x,
            " AND (role_id = 1 OR role_id = 2)
            ORDER BY nr_order DESC LIMIT 5")))
result = do.call(rbind, top5_R)
```

Question 8

Plot the overall number of movies in each year over time, and for each genre

GROUP BY year and ORDER BY year.

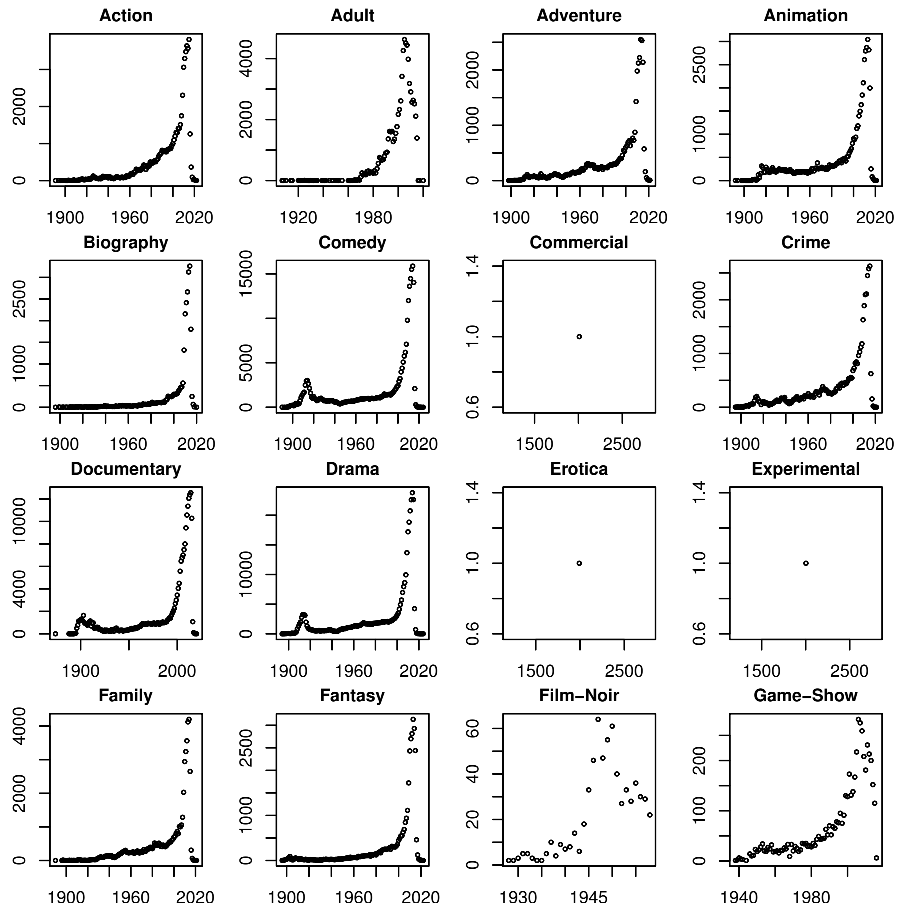
```
> q8sql = "SELECT COUNT(DISTINCT movie_id) AS number, production_year AS year
+ FROM title, movie_info, info_type
+ WHERE movie_info.info_type_id = info_type.id AND title.id = movie_info.movie_id
+ AND movie_info.movie_id = title.id AND kind_id = 1 AND info_type.info = 'genres'
+ GROUP BY production_year
+ ORDER BY production_year"
> year_num = dbGetQuery(conn, q8sql)
> plot(year_num$year, year_num$number, main = "Number of movies", xlab = "Year",
+ ylab = "Number", type = "l", cex = 1)
```

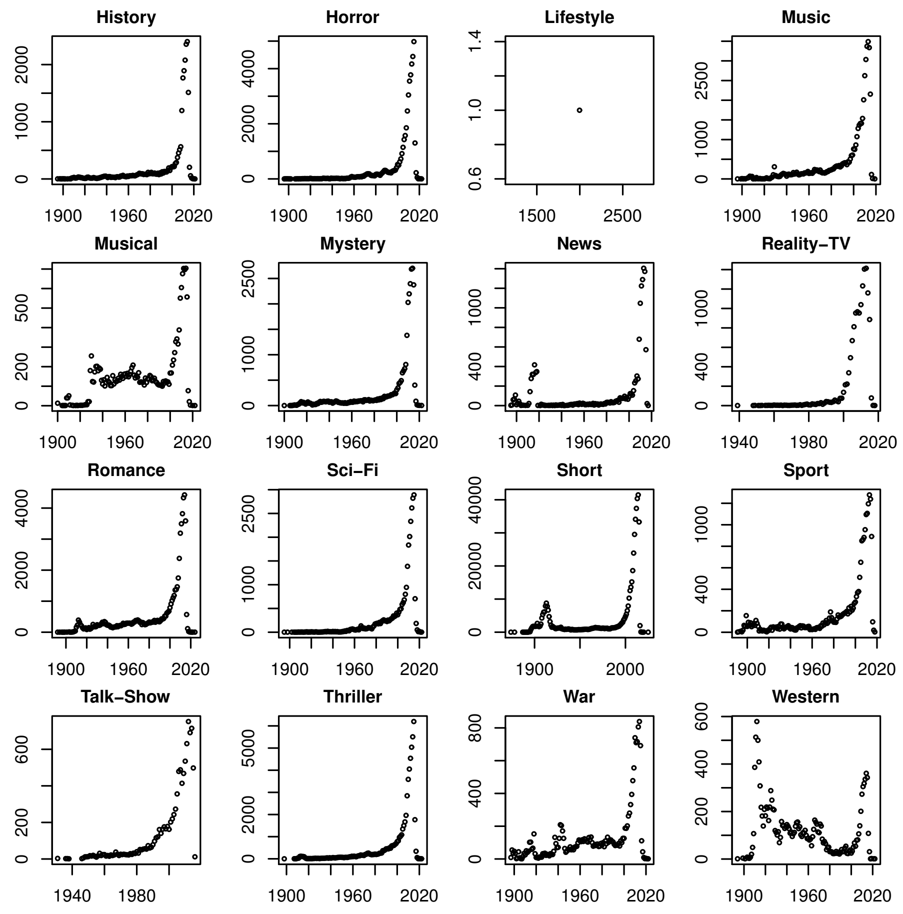


From this plot, we can find that the overall number of movies changed with time dramatically.

Group multiple columns.

```
> q8sql = "SELECT movie_info.info, production_year, COUNT(DISTINCT movie_id) AS number
+ FROM title, movie_info, info_type
+ WHERE movie_info.info_type_id = info_type.id AND title.id = movie_info.movie_id
+ AND movie_info.movie_id = title.id AND info_type.info = 'genres'
+ GROUP BY movie_info.info, production_year"
> group_year_info = dbGetQuery(conn, q8sql)
> year_info_split = split(group_year_info, group_year_info$info)
> par(mfrow = c(4,4), mar = c(1.9,1.9,1.9,1.9), cex.main = 0.8, cex.lab = 0.8, cex.axis = 0.8)
invisible(
  sapply(names(year_info_split), function(x)
        plot(year_info_split[[x]][["production_year"]], year_info_split[[x]][["number"]],
        main = x, xlab = "year", ylab = "number", cex = 0.5))
)
```


Action
Adult
Adventure
Animation
Biography
Comedy
Commercial
Crime
Documentary
Drama
Erotica
Experimental
Family
Fantasy
Film-Noir
Game-Show
0
1000
2000
2500
4000
5000
10000
15000
0.6
1.0
1.4
20
40
60
100
200
1500
2500
1900
1920
1930
1940
1945
1960
1980
2000
2020



For most genres, the number of movies changed with time dramatically.

Question 9

Who are the actors that have been in the most movies? List the top 20.

Using the table "movie_cast_name", we can do this question easily.

```
> q9sql = "SELECT person_id, name, COUNT(DISTINCT movie_id) AS number FROM movie_cast_name
+ GROUP BY person_id
+ ORDER BY COUNT(DISTINCT movie_id) DESC LIMIT 20"
> dbGetQuery(conn, q9sql)
   person_id              name number
1     195959         Blanc, Mel   1066
2     233482       Brahmanandam    987
3    1509290  Onoe, Matsunosuke    927
4    2615064       Flowers, Bess    831
5     801783       Hack, Herman    678
6    1585921         Phelps, Lee    650
7     969854        Jeremy, Ron    637
8     382758       Cobb, Edmund    634
9    1487999    O'Connor, Frank    625
10   1016068      Kapoor, Shakti    622
11    831924         Harris, Sam    616
12   1199644         London, Tom    609
13   1414125         Mower, Jack    594
14   1516542        Osborne, Bud    590
15   1362632      Miller, Harold    585
```

```
16    611864   Farnum, Franklyn    576
17    695721      Garcia, Eddie    566
18    180109       Bhasi, Adoor    563
19   1923163 Sreekumar, Jagathi    560
20   1693188   Richardson, Jack    559
```

Only using R

```
> cast_info = dbGetQuery(conn, "SELECT * FROM cast_info")
> title = dbReadTable(conn, "title")
> name = dbReadTable(conn, "name")
> cast_info$kind_id = title$kind_id[cast_info$movie_id]
> cast_info$year = title$production_year[cast_info$movie_id]
> cast_info$title = title$title[cast_info$movie_id]
> cast_info$name = name$name[cast_info$person_id]
> cast_info = cast_info[cast_info$kind_id == 1 & (cast_info$role_id == 1 | cast_info$role_id == 2),]
> dim(cast_info)
[1] 7033750      11
> person_movie = cast_info[, c("person_id","movie_id")]
> person_movie = person_movie[!duplicated(person_movie), ]
> dim(person_movie)
[1] 6938753       2
> sort(table(person_movie$person_id), decreasing = TRUE)[1:20]

 195959  233482 1509290 2615064  801783 1585921  969854  382758 1487999 1016068
   1066     987     927     831     678     650     637     634     625     622
831924 1199644 1414125 1516542 1362632  611864  695721  180109 1923163 1693188
 616     609     594    590     585     576     566     563     560     559
```

The names of this vector are our person_id, and the value of this vector are the number of movies. The result is exactly the same as SQL.

Question 10

Who are the actors that have had the most number of movies with "top billing", i.e., billed as 1, 2 or 3?

Define the actors with top billing as those actors with nr_order =1, 2 or 3. Count on the condition of nr_order.

Select top10.

```
> q10sql = "SELECT person_id, name, COUNT(DISTINCT movie_id) AS number,
+ MAX(production_year) AS max_year, MIN(production_year) AS min_year
+ FROM movie_cast_name
+ WHERE nr_order >= 1 AND nr_order <= 3
+ GROUP BY person_id
+ ORDER BY COUNT(DISTINCT movie_id) DESC LIMIT 10"
> dbGetQuery(conn, q10sql)
   person_id                                name number max_year min_year
1     195959                          Blanc, Mel    473     2011     1944
2    1856461                       Shin, Sung-il    394     1992     1960
3    1042765                 Kerrigan, J. Warren    370     1934     1910
4    1397573                          Moran, Lee    368     1933     1912
5    1223506                        Lyons, Eddie    354     1924     1911
6      54832 Anderson, Gilbert M. 'Broncho Billy'    320     1922     1904
```

```
7     825587                        Hardy, Oliver    311     1982     1914
8    1608412                      Pollard, 'Snub'    301     1933     1915
9    1693188                     Richardson, Jack    294     1929     1911
10    695721                        Garcia, Eddie    292     2013     1953
```

Only using R

```
> R_top10_billing = cast_info[!is.na(cast_info$nr_order) & cast_info$nr_order>=1
+ & cast_info$nr_order<=3,]
> #Because the id in title is exactly the same to its index,
> #so we can match R_top10_billing and title easily.
> R_top10_billing$year = title$production_year[R_top10_billing$movie_id]
> R_top10_billing = R_top10_billing[!duplicated(R_top10_billing[,c("person_id","movie_id")]),]
> dim(R_top10_billing)
[1] 879462     11
> sort(table(R_top10_billing$person_id), decreasing = TRUE)[1:10]

 195959  1856461 1042765 1397573 1223506   54832  825587 1608412 1693188  695721
    473     394     370     368     354     320     311     301     294     292




> top10 = sort(table(R_top10_billing$person_id), decreasing = TRUE)[1:10]
> top10_id = names(top10)

> top10_year = lapply(top10_id, function(x)
                    c(min(R_top10_billing[R_top10_billing$person_id == as.numeric(x),"year"]),
                    max(R_top10_billing[R_top10_billing$person_id == as.numeric(x),"year"]))
      )
> names(top10_year) = top10_id
> top10_year
$`195959`
[1] 1944 2011

$`1856461`
[1] 1960 1992

$`1042765`
[1] 1910 1934

$`1397573`
[1] 1912 1933

$`1223506`
[1] 1911 1924

$`54832`
[1] 1904 1922

$`825587`
[1] 1914 1982

$`1608412`
[1] 1915 1933

$`1693188`
```

```
[1] 1911 1929

$`695721`
[1] 1953 2013
```

Get the same answer as SQL.
Question 11

Who are the 10 actors that performed in the most movies within any given year?

10 actors that performed in the most movies within any given year.

```
> q11sql = "SELECT person_id, name, production_year AS year, COUNT(DISTINCT movie_id) AS number
+ FROM movie_cast_name
+ GROUP BY person_id, production_year
+ ORDER BY COUNT(DISTINCT movie_id) DESC LIMIT 10"
> top10 = dbGetQuery(conn, q11sql)
> top10
   person_id                name year number
1    1833458       Sennett, Mack 1909    125
2     977755  Johnson, Arthur V. 1909    116
3     127463    Barnett, Chester 1913    105
4    3452431         White, Pearl 1913    104
5    1394456          Moore, Owen 1909    102
6    1042765 Kerrigan, J. Warren 1912     99
7    1509290   Onoe, Matsunosuke 1918     92
8    1042765 Kerrigan, J. Warren 1911     86
9    1509290   Onoe, Matsunosuke 1915     86
10   1509290   Onoe, Matsunosuke 1914     84
```

What are their names, the year they starred in these movies and the names of the movies?

To get all the movies for each person in each year, we need to construct a temporary table, this will make some convenience to my computation.

```
dbSendQuery(conn, "CREATE TEMPORARY TABLE top10 AS
                   SELECT person_id, name, production_year AS year,
                   COUNT(DISTINCT movie_id) AS number
                   FROM movie_cast_name
                   GROUP BY person_id, production_year
                   ORDER BY COUNT(DISTINCT movie_id) DESC LIMIT 10")
```

Select all movies those top10 actors in each year.

```
> q11sql = "SELECT DISTINCT movie_cast_name.person_id, movie_cast_name.name,
+ production_year AS year, title FROM movie_cast_name, top10
+ WHERE movie_cast_name.person_id = top10.person_id
+ AND movie_cast_name.production_year = top10.year"
> all_movies = dbGetQuery(conn, q11sql)
> head(all_movies)
  person_id             name year                 title
1    127463 Barnett, Chester 1913   A Bachelor's Finish
2    127463 Barnett, Chester 1913      A Call from Home
3    127463 Barnett, Chester 1913   A Child's Influence
4    127463 Barnett, Chester 1913    A Dip Into Society
5    127463 Barnett, Chester 1913         A Hidden Love
6    127463 Barnett, Chester 1913 A Joke on the Sheriff
```

```
> dim(all_movies)
[1] 999   4
> sum(top10$number)
[1] 999
```

The number of movies we extract is exactly the same as the sum of the number of the top10 actors' movies.

Only using R
In this question, the big data set cost me really a long long time to compute, so in this question, I used the data with first 100,000 rows. Of course, this will cause the difference between the result of R and SQL.

```
> cast_info_distinct = cast_info[!duplicated(cast_info[, c("person_id", "movie_id")]), ]
> cast_info_distinct = cast_info_distinct[!is.na(cast_info_distinct$year),]
> cast_info_distinct$id_year = paste("person_id:", as.character(cast_info_distinct$person_id), "name:",
+ cast_info_distinct$name, "year:",as.character(cast_info_distinct$year))
> sort(table(cast_info_distinct$id_year), decreasing = TRUE)[1:10]

 person_id: 9932 name: Adair, Robyn year: 1915 person_id: 11211 name: Adams, Ernie year: 1938
                                            41                                            34
person_id: 11211 name: Adams, Ernie year: 1946  person_id: 9620 name: Acuff, Eddie year: 1940
                                            32                                            32
person_id: 11211 name: Adams, Ernie year: 1937 person_id: 11211 name: Adams, Ernie year: 1941
                                            30                                            29
 person_id: 9620 name: Acuff, Eddie year: 1939  person_id: 9620 name: Acuff, Eddie year: 1942
                                            27                                            27
 person_id: 9932 name: Adair, Robyn year: 1916 person_id: 11211 name: Adams, Ernie year: 1939
                                            27                                            26
> #Their names are included in the previous table

> top10 = sort(table(cast_info_distinct$id_year), decreasing = TRUE)[1:10]
> top10_id_str = strsplit(names(top10)," ")
> #Get their id
> top10_id = sapply(top10_id_str, function(x) as.numeric(x[2]))
> #Get their year
> top10_year = sapply(top10_id_str, function(x) as.numeric(x[length(x)]))
> result = cast_info_distinct[cast_info_distinct$id_year %in% names(top10), c("person_id",
+ "name", "year", "title")]
> result = result[order(result$person_id, result$year), ]
> dim(result)
[1] 305   4
> head(result)
      person_id         name year                  title
56519      9620 Acuff, Eddie 1939                 Ambush
56525      9620 Acuff, Eddie 1939               Backfire
56538      9620 Acuff, Eddie 1939            Blind Alley
56543      9620 Acuff, Eddie 1939 Blondie Meets the Boss
56572      9620 Acuff, Eddie 1939    Days of Jesse James
56583      9620 Acuff, Eddie 1939        Espionage Agent
```

Question 12

Who are the 10 actors that have the most aliases (i.e., see the aka_names table).

Select the number of aliases who are movie actors and limit to top10.

```
> q12sql = "SELECT person_id, name,
+ COUNT(DISTINCT name) AS number_aliases
+ FROM aka_name
+ WHERE person_id IN
+ (SELECT DISTINCT person_id FROM movie_cast_name)
+ GROUP BY person_id
+ ORDER BY COUNT(DISTINCT name) DESC LIMIT 10"
> dbGetQuery(conn, q12sql)
   person_id                  name number_aliases
1     662453          Franco, Jess             78
2     444281           D'Amato, Joe             70
3    2543347          Digard, Uschi             62
4    1796694       Savage, Herschel             53
5     882821            Ho, Godfrey             50
6    1869225          Silvera, Joey             42
7     373754       Clark, Christoph             37
8    1098131 Kronos, Donald Arthur             37
9    1176039        Len, Nathanael             37
10   3154545        Presova, Zuzana             37
```

Only using R

```
> aka_name = dbReadTable(conn, "aka_name")
> cast_info_full = dbReadTable(conn, "cast_info")
> actor_id = unique(cast_info_full$person_id[cast_info_full$role_id %in% c(1,2)])

> aka_name_actors = aka_name[aka_name$person_id %in% actor_id, ]
> aka_name_actors$actual_name = name$name[aka_name_actors$person_id]
> aka_name_actors = aka_name_actors[!duplicated(aka_name_actors[, c("person_id","name")]),]
> dim(aka_name_actors)
[1] 705912      9
> sort(table(aka_name_actors$person_id), decreasing = TRUE)[1:10]

 662453  444281 2543347 1796694  882821 1869225  373754 1098131 1176039 3154545
     78      70      62      53      50      42      37      37      37      37
```

The result is exactly the same as the result of SQL.

Question 13: Networks

Find our lead actor first.

```
> q13sql = "SELECT person_id, name, COUNT(DISTINCT movie_id) FROM movie_cast_name
+ GROUP BY person_id
+ HAVING COUNT(DISTINCT movie_id) > 20
+ ORDER BY COUNT(DISTINCT movie_id) ASC LIMIT 10"
> dbGetQuery(conn, q13sql)
   person_id                  name COUNT(DISTINCT movie_id)
1       5973 Abraham-Kremer, Bruno                       21
2       7187     Abu-Warda, Yussuf                       21
3       8455          Achorn, John                       21
4      14291          Adkins, Willy                       21
5      14477           Adler, Jerry                       21
6      18116        Aguilar, Adolfo                       21
7      22683       Ajaye, Franklyn                       21
8      22698              Ajaykumar                       21
9      23010         Akai, Hidekazu                       21
10     24143     Akinnagbe, Gbenga                       21
```

We choose the actor with person_id '5973' as our lead actor.

```
> lead_id = 5973
> lead_name = "Abraham-Kremer, Bruno"
```

Find the actor who has been in the same movie with '5973', to limit the number of these actors, we will only choose those with nr_order ¡ 3.
Mention that Duncan recommand us to use nr_order to reduce actors after extracting all of them, however I would like to use nr_order to reduce actors during the extraction, because if we do it after extraction, it is possible that we will get some actors with no connection to any other actors - who is isolated.

```
> q13sql = "SELECT person_id, name, nr_order FROM movie_cast_name
+ WHERE movie_id IN
+ (SELECT DISTINCT movie_id FROM movie_cast_name
+ WHERE person_id = 5973) AND (person_id = 5973 OR nr_order < 3)"
> net_person_id_list = dbGetQuery(conn, q13sql)
> #Record the person_id we select
> person_id_list = net_person_id_list$person_id
> person_id_list = paste(person_id_list, collapse = ", ")
> length(unique(net_person_id_list$person_id))
[1] 35
> #We have 35 actors now
> 5973 %in% net_person_id_list$person_id
[1] TRUE
```

Find the actors that has been in the same movie with those actors in our 'net_person_id_list', to limit the number of total actors in our network, we will only choose those with nr_order ¡ 2

```
> q13sql = paste("SELECT person_id, name, movie_id, nr_order FROM movie_cast_name
+ WHERE movie_id IN
+ (SELECT DISTINCT movie_id FROM movie_cast_name
+ WHERE person_id IN (", person_id_list, "))
+ AND (person_id IN (", person_id_list, ") OR nr_order < 2)")
> net_person_id_list_full = dbGetQuery(conn, q13sql)
> length(unique(net_person_id_list_full$person_id))
> actor_id = paste(net_person_id_list_full$person_id, collapse = ", ")
> q13sql = paste("SELECT DISTINCT person_id, name, movie_id, nr_order FROM movie_cast_name
              WHERE person_id IN (",actor_id, ")")
> net_person_id_list_full = dbGetQuery(conn, q13sql)
> length(unique(net_person_id_list_full$person_id))
[1] 674
```

Now we finally get 674 actors in our network.
Initialize our network matrix.

```
> n = length(unique(net_person_id_list_full$person_id))
> net = matrix(0, n, n)
> length(unique(net_person_id_list_full$name))
[1] 674
```

The length of unique name is exactly the same as the length of id, so we can use name to set the matrix's colnames and rownames.

```
> full_name_list = unique(net_person_id_list_full$name)
> colnames(net) = full_name_list
> rownames(net) = full_name_list
```

Combination, get the pairs of all actors

```
> comb_name = combn(full_name_list, 2)
> comb_name = as.data.frame(t(comb_name))
> colnames(comb_name) = c("actor1","actor2")
> comb_name$actor1 = as.character(comb_name$actor1)
> comb_name$actor2 = as.character(comb_name$actor2)
> dim(comb_name)  #equals to choose(674,2)
[1] 226801      2
> head(comb_name)
             actor1                 actor2
1 Abatantuono, Diego           Abbasi, Riz
2 Abatantuono, Diego     Abelanski, Lionel
3 Abatantuono, Diego Abraham-Kremer, Bruno
4 Abatantuono, Diego          Adam, Alfred
5 Abatantuono, Diego            Adams, Kev
6 Abatantuono, Diego           Allio, Paul
```

Construct a new column, if the actors in the 1st and 2nd have been in a same movie, then the value of the new column is TRUE, otherwise FALSE.

```
net_person_name_split = split(net_person_id_list_full, net_person_id_list_full$name)
```

Design a function to do so.

```
> if_related = function(actor1, actor2, net_person_name_split){
+ #Extract corresponding movie_id
+ movie_actor1 = net_person_name_split[[actor1]][["movie_id"]]
+ movie_actor2 = net_person_name_split[[actor2]][["movie_id"]]
+ if(length(intersect(movie_actor1, movie_actor2)) == 0){   #their intersection.
+ return(FALSE)
+ }
+ else{
+ return(TRUE)
+ }
+ }
```

Construct the new column.

```
> comb_name$relate = mapply(function(x, y) if_related(x, y, net_person_name_split),
+ comb_name$actor1, comb_name$actor2)
> head(comb_name)
             actor1                 actor2 relate
1 Abatantuono, Diego           Abbasi, Riz  FALSE
2 Abatantuono, Diego     Abelanski, Lionel  FALSE
3 Abatantuono, Diego Abraham-Kremer, Bruno  FALSE
4 Abatantuono, Diego          Adam, Alfred  FALSE
5 Abatantuono, Diego            Adams, Kev  FALSE
6 Abatantuono, Diego           Allio, Paul  FALSE
```

Only get the related pairs.

```
comb_name_relate = comb_name[comb_name$relate,]
```

Update the network matrix.

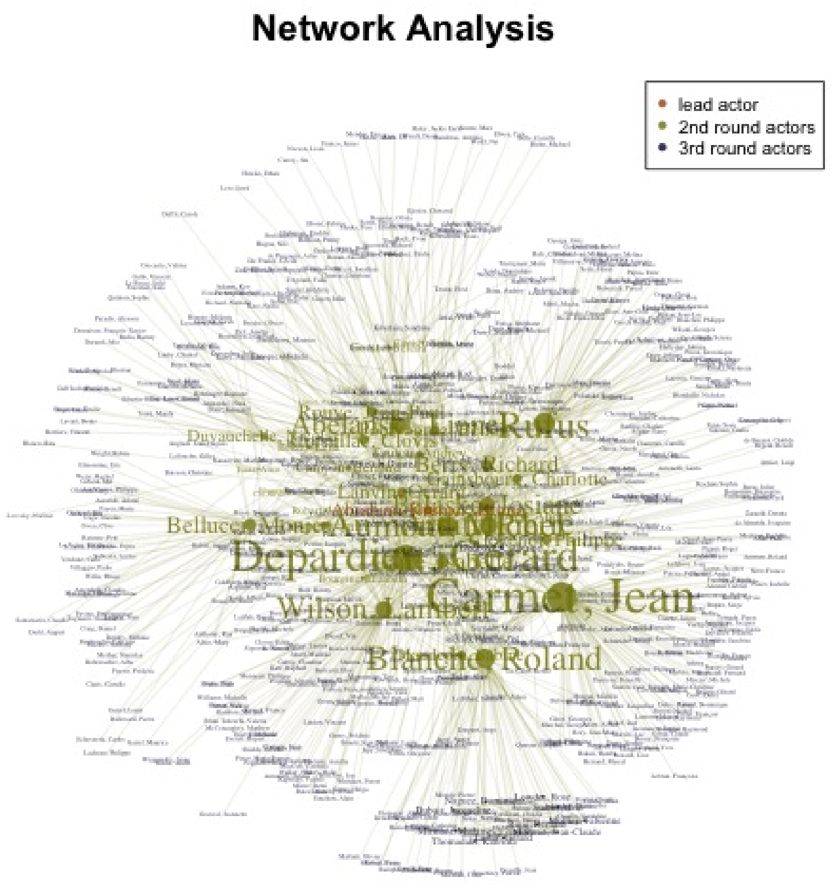
```
> for(i in 1:nrow(comb_name_relate)){
+ net[comb_name_relate$actor1[i],comb_name_relate$actor2[i]] = 1
+ net[comb_name_relate$actor2[i],comb_name_relate$actor1[i]] = 1
+ }
```

I try to use sapply and assgin function to change global environment - the matrix, it cost a long time and the R session aborted.
If I use for() loop, R can do it successfully within 1 second. So I just use the for() loop.
So we have successfully constructed our network matrix.
Then we are going to plot the network graph by using 'grap' in 'igraph' package.
I get the information about igraph from "http://www.rdatamining.com/examples/social-network-analysis"

```
#network graph, mainly from Google.
> par(mfrow = c(1,1), mar = c(3,3,3,3))
> g = graph.adjacency(net, mode = "undirected")
> g = simplify(g)
> V(g)$label = V(g)$name
> V(g)$degree = degree(g)

> V(g)$label.cex = 1.*V(g)$degree/max(V(g)$degree)+ 0.3
> #Assign different colors to the lead actor and actors related to him/her.
> color_vector = rep(rgb(0, 0, 0.2, 0.8), nrow(net))
> lead_index = which(names(V(g)) == lead_name)
> round2_name = unique(net_person_id_list$name)
> round2_index = which(names(V(g)) %in% round2_name & names(V(g)) != lead_name)

> color_vector[round2_index] = rgb(0.5, 0.5, 0, 0.8)
> color_vector[lead_index] = rgb(0.7, 0.3, 0, 0.8)
> V(g)$label.color = color_vector
> V(g)$frame.color = NA
> E(g)$color = rgb(0.5, 0.5, 0, 0.5)
> E(g)$width = 0.3
> vertex_size = 8*V(g)$degree/max(V(g)$degree) + 0.1
> # plot the graph in layout1
> #plot(g, layout=layout1, vertex.size = vertex_size, main = "Network Analysis")
> plot(g, layout=layout.kamada.kawai, vertex.size = vertex_size,
  vertex.color = color_vector, main = "Network Analysis")
> legend("topright", legend = c("lead actor", "2nd round actors", "3rd round actors"),
col = c(rgb(0.7, 0.3, 0, 0.8), rgb(0.5, 0.5, 0, 0.8), rgb(0, 0, 0.2, 0.8)),
        pch = 16, cex = 0.6)
```



From this network graph, we can find that most actors has few connections, and actors at the center of the network have a lot of connections with other actors.

Question 14

What are the 10 television series that have the most number of movie stars appearing in the shows?

First we have the define the concept of ”movie stars”.
We Define ”movie stars” as those actors who has played a movie with nr_order ¡ 5.

```
> dbGetQuery(conn, "SELECT * FROM kind_type")
  id           kind
1  1          movie
2  2      tv series
3  3       tv movie
4  4    video movie
5  5 tv mini series
6  6     video game
7  7        episode
```

kind_id = 2 equals to tv series.
First get the person_id of all ”movie stars”, and then select tv series which has most of these ”movie stars”.

```
> q14sql = "SELECT movie_id, title.title, COUNT(DISTINCT person_id) AS number
+ FROM title, cast_info
+ WHERE title.id = cast_info.movie_id
+ AND kind_id = 2 AND person_id IN
+ (SELECT DISTINCT person_id FROM movie_cast_name
+ WHERE nr_order < 5)
+ GROUP BY movie_id
+ ORDER BY COUNT(DISTINCT person_id) DESC LIMIT 10"
> dbGetQuery(conn, q14sql)
   movie_id                    title number
1    729678         General Hospital    527
2   1404883         One Life to Live    388
3    449619        Days of Our Lives    371
4    122527            Another World    336
5   1941002         The Guiding Light    318
6     76799           All My Children    257
7    147610        As the World Turns    250
8   1970321 The Laurel and Hardy Show    244
9   1917967         The Edge of Night    208
10  1572213     Retrosexual: The 80's    196
```