STA141 Assignment V

Weitong(Jessie) Lin

November 23, 2015

pre-load dataset

```
# load the package
library(RSQLite)
```

```
## Loading required package: DBI
```

```
# connect to the imdb database
imdb = dbConnect(SQLite(), '~/Desktop/lean_imdbpy.db')
# list all tables
dbListTables(imdb)
```

```
##  [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"
```

For this assignment, I choose to use 8.48 gigabyte dataset to do the 1-9 questions. The results above are the table names under this database. Before starting to answer the questions, I first try to explore this data set in UNIX Shell.

First, I connect to this database 'lean_imdbpy'

```
sqlite3 Desktop/lean_imdbpy.db
```

Then I see the structure of this database and their corresponding primary key.

```
.schema
```

After looking around this database, I have a general idea abou the structure of it. Now let's answer each quastion.

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

- **How many actors**

When talking about "actor", I first try to look at all possible tables which are related to names of entertainment industry staffs. For this question, I think the tables *cast_info*, *name* and *role_type* can be linked together by their relative primary keys to get a lot of information of actors.

Before looking at *role_type* table, I thought that all staffs list in *name* and *cast_info* are about actors. However, let's see the *role_type* table:

```
dbGetQuery(imdb,"SELECT * FROM role_type;")
```

```
##    id                role
## 1   1               actor
## 2   2             actress
## 3   3            producer
## 4   4              writer
## 5   5     cinematographer
## 6   6            composer
## 7   7    costume designer
## 8   8            director
## 9   9              editor
## 10 10  miscellaneous crew
## 11 11 production designer
## 12 12               guest
```

Not only actors include in this database, but also the all possible occupations in this industry are covered.

Thus, we can see how many distinct *person_id* for *role_type* are 'actor' and 'actress'exists. For the consideration of time-consuming, I only use *cast_info* and *role_type* to get the answer in this question.

```
#########
# Here we join theses two tables:
#     cast_info.role_id & role_type.id
#########

actor_num = dbGetQuery(imdb, 'SELECT COUNT(DISTINCT person_id)
                              FROM cast_info, role_type
                              WHERE role_id = role_type.id
                              AND role IN ("actor", "actress");')
actor_num
```

```
##   COUNT(DISTINCT person_id)
## 1                   3492018
```

Here we can see that there are 3492018 actors in the database.

- **How many movies**

First, I look at what type we have:

```
dbGetQuery(imdb,"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
```

Here we can see that there are 7 types. Since we want to know how many movies in the database, I will only choose 'movie', 'tv movie' and 'video movie' as the movies.

```
#########
# Here we join these two tables:
#     title.kind_id & kind_type.id.
# title.id is the identity for each record
# we choose only the type which has string 'movie' in it.
#########
dbGetQuery(imdb,"SELECT COUNT(title.id)
           FROM title, kind_type
           WHERE kind_id = kind_type.id
           AND kind LIKE '%movie%';")
```

```
##   COUNT(title.id)
## 1         1145814
```

There are 1145814 movies in the database.

2. What time period does the database cover?

```
#########
# We see the minimum and maximum year for all productions
#########
dbGetQuery(imdb,"SELECT MIN(production_year), MAX(production_year) FROM title;")
```

```
##   MIN(production_year) MAX(production_year)
## 1                 1874                 2025
```

The time period covers in this database is from 1874 to 2025. This time period makes me thinking that whether it's typo or not for those year is larger than 2015. Thus, I will take *year = 2025* as an example:

```
dbGetQuery(imdb,"SELECT title FROM title WHERE production_year = 2025;")
```

```
##           title
## 1 StreetDance 4
```

We can see that the movie whose production year is 2025, called *StreetDance 4*, will be indeed produced in 2025. Thus, this database may contain all productions which have already announced that they will be shot in the future. So this time period is reliable.

3. What proportion of the actors are female, male?

I found that there are two variables which are related to gender: *role_type.role* and *name.gender*. I'm not sure which one I should choose to identify the gender for actors. Thus, I will see whether there are some actors having diffenent identification result from these two variables.

```
# define actor type
actor_type = c("actor", "actress")
# define gender type
gender = c("f", "m")

#########
# Here we join theses three tables:
#     cast_info.person_id & name.id,
#     cast_info.role_id & role_type.id
# use sprintf function to get sql commands that we want
#########
sql_sen = sprintf("SELECT COUNT(DISTINCT person_id)
          FROM cast_info, name, role_type
          WHERE person_id = name.id
          AND role_id = role_type.id
          AND role = '%s'
          AND gender = '%s';", actor_type, gender)

# apply each sql command
diff_identity = sapply(sql_sen, function(sql_sen){

    # apply each sql command:
    #   - number of people: "actor" & "f"
    #   - number of people: "actress" & "m"

  dbGetQuery(imdb, sql_sen)
})
# name the list
names(diff_identity) = c('actor_f', 'actress_m')
diff_identity
```

```
## $actor_f
## [1] 0
##
## $actress_m
## [1] 1
```

Then we can see that there is a people whose gender is "male" but it's also an "actress". This situation is abnormal. Let's see who is this people:

```
dbGetQuery(imdb, 'SELECT DISTINCT title, name, gender, role, person_id
           FROM cast_info, name, role_type, title
           WHERE cast_info.person_id = name.id
           AND cast_info.role_id = role_type.id
           AND cast_info.movie_id = title.id
           AND role = "actress"
           AND gender = "m";')
```

```
##                         title           name gender    role person_id
## 1              Bad Luck Brian Edwards, Chris      m actress    568592
## 2 Birds Are Afraid of Heights Edwards, Chris      m actress    568592
## 3                   Boomerang Edwards, Chris      m actress    568592
```

```
## 4                 It's a Fap! Edwards, Chris      m actress    568592
## 5             Matrix Morpheus Edwards, Chris      m actress    568592
## 6                    Oh Fuck! Edwards, Chris      m actress    568592
## 7                    Pedobear Edwards, Chris      m actress    568592
```

From the result, we can see that this *Chirs Edwards* gets *male* and *actress* problem.

```
dbGetQuery(imdb, 'SELECT DISTINCT title, name, gender, role, kind
           FROM cast_info, name, role_type, title, kind_type
           WHERE cast_info.person_id = name.id
           AND cast_info.role_id = role_type.id
           AND cast_info.movie_id = title.id
           AND title.kind_id = kind_type.id
           AND person_id = 568592;')
```

```
##                          title           name gender    role    kind
## 1           Taste the Rainbow Edwards, Chris      m   actor episode
## 2              Bad Luck Brian Edwards, Chris      m actress episode
## 3 Birds Are Afraid of Heights Edwards, Chris      m actress episode
## 4                   Boomerang Edwards, Chris      m actress episode
## 5                It's a Fap! Edwards, Chris      m actress episode
## 6             Matrix Morpheus Edwards, Chris      m actress episode
## 7                    Oh Fuck! Edwards, Chris      m actress episode
## 8                    Pedobear Edwards, Chris      m actress episode
```

I search this person on Google then I find that all these 8 TV episodes belong to an animation called *Animeme*. Thus, I believe that *Chirs Edwards* is a voice cast who can dub for either male or female. Thus, *Chirs Edwards* is indeed a *male*, but has been either an *'actor'* or *'actress'* as a voice cast for an animated character.

Now I know that we had better not use *role_type.role* to define the gender identiciation for an actor. *name.gender* is a better choice.

```
# define gender type
gender = c("f", "m")

#########
# Here we join theses three tables:
#     cast_info.person_id & name.id,
#     cast_info.role_id & role_type.id
# use sprintf function to get sql commands that we want
#########
sql_sen1 = sprintf("SELECT COUNT(DISTINCT person_id)
          FROM cast_info, name, role_type
          WHERE person_id = name.id
          AND role_id = role_type.id
          AND role IN ('actor', 'actress')
          AND gender = '%s';", gender)

# apply each sql command
gender_num = sapply(sql_sen1, function(sql_sen){

    # apply each sql command:
    #   - number of people: "f"
```

```
    #   - number of people: "m"

  dbGetQuery(imdb, sql_sen)
})

# name the list
names(gender_num) = c('female', 'male')
gender_num
```

```
## $female
## [1] 1235134
##
## $male
## [1] 2256884
```

```
# Proportion
unlist(gender_num)/unlist(actor_num)
```

```
##    female      male
## 0.3537021 0.6462979
```

Here we can see that the proportion of the actors are female is 35.37%, male is 64.63%.

gender	female	male
prop	35.37%	64.63%

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

From Q1.2 we already know that there are 7 types. So we can directly count how many movies ar there for each kind_type:

```
#########
# Here we join these two tables:
#     title.kind_id & kind_type.id.
# title.id is the identity for each record
# we count the number of works for each kind
#########
num_kind = dbGetQuery(imdb,"SELECT kind, COUNT(title.id)
                            FROM title, kind_type
                            WHERE kind_id = kind_type.id
                            GROUP BY kind;")
colnames(num_kind) = c("kind_type", "count")

# calculate the proportion
prop_kind = num_kind
prop_kind[,2] = num_kind[,2]/sum(num_kind[,2])
# sort by the decreasing order
prop_kind = prop_kind[order(prop_kind[,2], decreasing = TRUE),]
prop_kind
```

```
##     kind_type       count
## 1     episode 0.635583712
## 2       movie 0.249111894
## 6 video movie 0.041563815
## 4   tv series 0.035273371
## 3    tv movie 0.034126175
## 5  video game 0.004341033
```

Here is a table to show the propotion of each kind type:

kind type	episode	movie	video movie	tv series	tv movie	video game
prop	63.56%	24.91%	4.16%	3.53%	3.41%	0.43%

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

For this question I will first restrict the denfinition of movie is 'movie', 'tv movie' and 'video movie' first. Then I use *movie_info* and *info_type* these two tables to find genres.

```
#########
# Here we join these two tables:
#      movie_info & info_type
# movie_info.info is the genre types when info_type.info  = "genres"
#########

# count how many genres
genres = dbGetQuery(imdb, "SELECT COUNT(DISTINCT movie_info.info)
                           FROM movie_info, info_type, title, kind_type
                           WHERE info_type_id = info_type.id
                           AND title.id = movie_info.movie_id
                           AND title.kind_id = kind_type.id
                           AND info_type.info  = 'genres'
                           AND kind LIKE '%movie%';")

# show their names
genres = dbGetQuery(imdb, "SELECT DISTINCT movie_info.info
                           FROM movie_info, info_type, title, kind_type
                           WHERE info_type_id = info_type.id
                           AND title.id = movie_info.movie_id
                           AND title.kind_id = kind_type.id
                           AND info_type.info  = 'genres'
                           AND kind LIKE '%movie%';")
genres
```

```
##           info
## 1       Comedy
## 2        Short
## 3        Adult
## 4        Drama
## 5    Animation
## 6         News
## 7      History
```

```
## 8           War
## 9        Horror
## 10    Adventure
## 11       Sci-Fi
## 12    Biography
## 13  Documentary
## 14       Family
## 15       Action
## 16      Romance
## 17      Musical
## 18        Sport
## 19      Fantasy
## 20      Mystery
## 21     Thriller
## 22        Music
## 23        Crime
## 24    Talk-Show
## 25   Reality-TV
## 26      Western
## 27    Game-Show
## 28    Film-Noir
## 29 Experimental
## 30   Commercial
## 31      Erotica
```

From the result, we can see that there are 31 genre types for this movie database. All names of genres are listed above.

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

In this question, I will consider the *movies* as what I've defined in the Q1.2, which contains 'movie', 'tv movie' and 'video movie'. Also, I will exact the number of movies for each genre from *movie_info* under *info_type* = 'genre':

```
#########
# Here we join these four tables:
#      movie_info, info_type, title, kind_type
# How I link them:
#      - movie_info.info is the genre types when info_type.info  = "genres"
#      - the link between title & kind_type can filter the only "movie" part
#      - the above two parts can be linked by title & movie_info
# count the number of movie for each genre and them sort them in descreasing order
# only show the top ten
#########

top10_genre = dbGetQuery(imdb, "SELECT movie_info.info, COUNT(title.id)
                                FROM movie_info, info_type, title, kind_type
                                WHERE info_type_id = info_type.id
                                AND title.id = movie_info.movie_id
                                AND title.kind_id = kind_type.id
                                AND info_type.info  = 'genres'
```

```
                              AND kind LIKE '%movie%'
                              GROUP BY movie_info.info
                              ORDER BY COUNT(title.id) DESC
                              LIMIT 10;")

colnames(top10_genre) = c('genre', 'num_movie')
top10_genre
```

```
##          genre num_movie
## 1        Short    519538
## 2        Drama    306901
## 3       Comedy    212813
## 4  Documentary    196893
## 5        Adult     71679
## 6     Thriller     58419
## 7      Romance     57719
## 8       Action     52412
## 9    Animation     45917
## 10      Horror     44670
```

Here we can see that movies with 'short', 'drama' and 'comedy' have the top 3 amount. And all other genres in top 10 are listed above.

7. 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?

Again, I will consider the *movies* as what I've defined in the Q1.2, which contains 'movie', 'tv movie' and 'video movie'.

- **How many are there?**

In this question we need to link the *movie_keyword* and *keyword* table to find the keyword **space**:

```
#########
# Here we join these four tables:
#      title, kind_type, movie_keyword, keyword
# How I link them:
#      - movie_keyword & keyword can be linked by the id
#      - the link between title & kind_type can filter the only "%movie%" part
#      - the above two parts can be linked by title & movie_keyword
# count the number of movie with keyword 'space'
#########

dbGetQuery(imdb,"SELECT COUNT(title.id)
                 FROM title, kind_type, movie_keyword, keyword
                 WHERE title.id = movie_keyword.movie_id
                 AND movie_keyword.keyword_id = keyword.id
                 AND title.kind_id = kind_type.id
                 AND kind LIKE '%movie%'
                 AND keyword LIKE 'space';")
```

```
##   COUNT(title.id)
## 1             534
```

Here we can see that there are 534 movies which have the keyword 'space'.

- **The years they release**

```
#########
# Here we join these four tables:
#      title, kind_type, movie_keyword, keyword
# How I link them:
#      - movie_keyword & keyword can be linked by the id
#      - the link between title & kind_type can filter the only "%movie%" part
#      - the above two parts can be linked by title & movie_keyword
# find the distinct year
# sort by year with a decreasing order
#########
years_movie = dbGetQuery(imdb,"SELECT DISTINCT production_year
                               FROM title, kind_type, movie_keyword, keyword
                               WHERE title.id = movie_keyword.movie_id
                               AND movie_keyword.keyword_id = keyword.id
                               AND title.kind_id = kind_type.id
                               AND kind LIKE '%movie%'
                               AND keyword LIKE 'space'
                               ORDER BY production_year DESC;")
```

From the result(since it's too long, I won't show it now), we find **NA** in our year. The reason is that some of movies has already settles the film crew but still not sure when this film will be released. Thus I will remove NA.

```
# remove NA
years_movie[!is.na(years_movie)]
```

```
##  [1] 2018 2017 2016 2015 2014 2013 2012 2011 2010 2009 2008 2007 2006 2005
## [15] 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 1993 1992 1991
## [29] 1990 1989 1988 1987 1986 1985 1984 1983 1982 1981 1980 1979 1978 1977
## [43] 1975 1974 1973 1972 1971 1970 1969 1968 1967 1966 1965 1964 1962 1961
## [57] 1960 1959 1958 1957 1956 1955 1954 1953 1951 1950 1947 1946 1930 1925
## [71] 1922 1918 1911
```

Here we can see that the space-related idea has already been introduced to moives since 1911. After 1946, which is exactly the year that the first photo from space was taken from a V-2 launched by US scientists [https://en.wikipedia.org/wiki/Space_exploration], space-related movies were almost released every year because people may be more curious about this mysterious space than before. Also, there are also some space-related movies will be released from 2016 to 2018.

- **top 5 actors in each of these movies**

cast_info.nr_order means the billing position, which means get the top paid in each movie. So here I will set *cast_info.nr_order* between 1 and 5 to subset the top 5 biling positions.

```
#########
# Here we join these seven tables:
#      title, kind_type, movie_keyword, keyword
# How I link them:
#      - movie_keyword & keyword can be linked by the id
#      - the link between title & kind_type can filter the only "%movie%" part
#      - cast_info, name, role_type are linked. Here we can choose "actors"
#      - the above three parts can be linked by title & movie_keyword, title & cast_info
# sort by title, we can see the top 5 actor for each movie
#########

top_5_each = dbGetQuery(imdb,"SELECT DISTINCT name, nr_order, title
                 FROM title, kind_type, movie_keyword, keyword, cast_info, name, role_type
                        WHERE title.id = movie_keyword.movie_id
                        AND movie_keyword.keyword_id = keyword.id
                        AND title.kind_id = kind_type.id
                        AND cast_info.person_id = name.id
                        AND cast_info.movie_id = title.id
                        AND role_id = role_type.id
                        AND role IN ('actor', 'actress')
                        AND kind LIKE '%movie%'
                        AND keyword LIKE 'space'
                        AND cast_info.nr_order BETWEEN 1 AND 5
                        ORDER BY title, nr_order;")
```

Since there are too many results which is hard to put all of them in my reports, I will show three movies *Interstellar*, *Guardians of the Galaxy*, *Avatar* that I love most and show their top 5 billing position actors.

```
# the movies I love
movie_name = c("Interstellar", "Guardians of the Galaxy", "Avatar")

example_movie = lapply(movie_name, function(movie) {

    # INPUT:
    #   - no value: insert "NA"


  top_5_each[top_5_each$title == movie, ]

})
names(example_movie) = movie_name

example_movie
```

```
## $Interstellar
##                     name nr_order        title
## 451       Burstyn, Ellen        1 Interstellar
## 452 McConaughey, Matthew        2 Interstellar
## 453       Foy, Mackenzie        3 Interstellar
## 454        Lithgow, John        4 Interstellar
## 455   Chalamet, Timothée        5 Interstellar
##
## $`Guardians of the Galaxy`
```

```
##                name nr_order                   title
## 404    Pratt, Chris        1 Guardians of the Galaxy
## 405    Saldana, Zoe        2 Guardians of the Galaxy
## 406  Bautista, Dave        3 Guardians of the Galaxy
## 407     Diesel, Vin        4 Guardians of the Galaxy
## 408 Cooper, Bradley        5 Guardians of the Galaxy
##
## $Avatar
##                    name nr_order  title
## 113    Worthington, Sam        1 Avatar
## 114        Saldana, Zoe        2 Avatar
## 115   Weaver, Sigourney        3 Avatar
## 116       Lang, Stephen        4 Avatar
## 117 Rodriguez, Michelle        5 Avatar
```

From the above result, we can see the top 5 actors for my favourite movies.

8. 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.

Now let get the number of movies for each genre over years. From the previous question, we know that some unfinished movies haven't decide the releasing year, which are marked as "NA" in *production_year*. I will remove them.

```
#########
# Here we join these four tables:
#      movie_info, info_type, title, kind_type
# How I link them:
#      - the link between title & kind_type can filter the only "%movie%" part
#      - - movie_info.info is the genre types when info_type.info  = "genres"
#      - the above two parts can be linked by title & movie_info
# group table by genre and year
#########

genre_year_num = dbGetQuery(imdb,'SELECT movie_info.info, production_year,
                                        COUNT(DISTINCT title.id)
                            FROM movie_info, info_type, title, kind_type
                            WHERE info_type_id = info_type.id
                            AND title.id = movie_info.movie_id
                            AND title.kind_id = kind_type.id
                            AND info_type.info  = "genres"
                            AND kind LIKE "%movie%"
                            GROUP BY movie_info.info, production_year;')
# name the list
names(genre_year_num) = c('genre', 'year', 'num_movie')

# remove 'NA' year
genre_year_num_no = genre_year_num[!is.na(genre_year_num$year),]

# show sample result
head(genre_year_num_no)
```

```
##    genre year num_movie
## 2 Action 1891         1
## 3 Action 1894         3
## 4 Action 1896         1
## 5 Action 1897         3
## 6 Action 1898         2
## 7 Action 1900         1
```

Now I will classify this 31 genres into 8 blocks by their similarity:

From question 9 to 12, I will use the new dataset *lean_imdbpy_2010_idx.db* because of time-consuming.Also, I will consider the *movies* as what I've defined in the Q1.2, which contains 'movie', 'tv movie' and 'video movie'.

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

- **load the new dataset**

```
imdb2 = dbConnect(SQLite(), '~/Desktop/lean_imdbpy_2010_idx.db')
# list all tables
dbListTables(imdb2)
```

```
##  [1] "aka_name2"       "aka_title2"      "cast_info2"
##  [4] "info_type"       "keyword2"        "kind_type"
##  [7] "movie_info2"     "movie_info_idx2" "movie_keyword2"
## [10] "name2"           "person_info2"    "role_type"
## [13] "sqlite_sequence" "title2"
```

- **SQL part**

Under the *movies actors* condition, I will count the distinct movie (which can be seen by *movie_id*) for each person. Since people may have the same name, *person_id* would be a better choice since it's unique for everyone.

```
#########
# Here we join these five tables:
#     name2, cast_info2, title2, role_type, kind_type
# How they link
#     - name2, cast_info2 and role_type can be linked to get actor info
#     - title2 & kind_type to get the movies type
#     - above two parts can be linked by cast_info2 & title2
# get the name and the number of movie
#########
actor_movie_sql = dbGetQuery(imdb2, 'SELECT person_id, name,
                   COUNT(DISTINCT cast_info2.movie_id) AS num_movie
                   FROM name2, cast_info2, title2, role_type, kind_type
                   WHERE role_id = role_type.id
```

```
                  AND cast_info2.movie_id = title2.id
                  AND cast_info2.person_id = name2.id
                  AND title2.kind_id = kind_type.id
                  AND role IN ("actor", "actress")
                  AND kind LIKE "%movie%"
                  GROUP BY person_id
                  ORDER BY num_movie DESC
                  LIMIT 20;')
actor_movie_sql
```

```
##    person_id                 name num_movie
## 1    1708783        Roberts, Eric       207
## 2    1025363       Kaufman, Lloyd       171
## 3    1494149        Obama, Barack       159
## 4     233482         Brahmanandam       146
## 5    1204854       Lorente, Txema       140
## 6    1705106        Rivers, Scott       131
## 7    3123037          Pell, Rhoda       129
## 8    1494693     Oberst Jr., Bill       120
## 9    2012005      Thingvall, Joel       119
## 10   3091849         Olsen, Maria       114
## 11   2046271         Trejo, Danny       113
## 12    506474        Dewdney, Paul       111
## 13    760186     Graf, David Alan       110
## 14   1736974         Rosete, Jose       105
## 15    969854          Jeremy, Ron       104
## 16    662433        Franco, James       103
## 17   1302470    Mazak, Kasey Ryne       102
## 18   3343788        Swift, Taylor       102
## 19     60190     Andrisan, Rodrig       100
## 20    831233 Harris, Lee Nicholas        97
```

From the result above, we can see that **Roberts, Eric** has the most movies which is 207 after 2010.

- **R part**

First I read some table which might be useful later on:

```
name2 = dbReadTable(imdb2,'name2')
cast_info2 = dbReadTable(imdb2,'cast_info2')
title2 = dbReadTable(imdb2,'title2')
role_type = dbReadTable(imdb2,'role_type')
kind_type = dbReadTable(imdb2,'kind_type')
```

Since we can not joint table in R, let's see what the corresponding id are for *role_type* and *kind_type*, which can be used in *cast_info2* and *title2*

```
# role_type for cast_info2
actor_id = role_type[role_type$role == c("actor", "actress"),'id']
# kind_type for title2
type_id = kind_type[kind_type$kind %in% c("movie", "tv movie", "video movie"),'id']
```

Thus, for what we interest at, we can subset the *cast_info2* and *title2* tables:

```
# subset cast_info2
cast_info2_actor = cast_info2[cast_info2$role_id %in% actor_id, ]
# sunset title2
title2_type = title2[title2$kind_id %in% type_id, ]
```

Although we use R to get the result, the basic idea is very similar to SQL. We should find their inner connection and then subset the dataset.

```
# create a link between cast_info2_actor and title2_type
movie_link = cast_info2_actor$movie_id %in% title2_type$id
# only leave the 'person_id','movie_id' for what we have subseted
actor_movie = cast_info2_actor[movie_link, c('person_id','movie_id')]
# remove the duplicated record
actor_movie = unique(actor_movie)
# count the number of movies for each person
num_movie  = as.data.frame(table(actor_movie$person_id))
# set the colnames
colnames(num_movie) = c('person_id','num_movie')


# create a link between name2 and num_movie
name_link = name2$id %in% num_movie$person_id
# now get the name of actor by the same 'person_id'
name_actor = name2[name_link, c('id','name')]
# set the colnames
colnames(name_actor) = c('person_id','name')


# now merge the above two table with the same column 'person_id'
actor_movie_r = merge(name_actor, num_movie)
# order the new table with num_movie in decreasing order
actor_movie_r = actor_movie_r[order(actor_movie_r[,3], decreasing = TRUE), ]
# only show the top 20
top20_actor_movie_r = head(actor_movie_r,20)
top20_actor_movie_r
```

```
##         person_id                 name num_movie
## 608124    1708783        Roberts, Eric       207
## 365105    1025363       Kaufman, Lloyd       171
## 531950    1494149        Obama, Barack       159
## 83424      233482         Brahmanandam       146
## 427531    1204854       Lorente, Txema       140
## 606801    1705106        Rivers, Scott       131
## 1078971   3123037          Pell, Rhoda       129
## 532140    1494693     Oberst Jr., Bill       120
## 715181    2012005      Thingvall, Joel       119
## 1068692   3091849         Olsen, Maria       114
## 727257    2046271         Trejo, Danny       113
## 181058     506474        Dewdney, Paul       111
## 271473     760186     Graf, David Alan       110
## 618220    1736974         Rosete, Jose       105
```

```
## 345190     969854          Jeremy, Ron       104
## 236451     662433        Franco, James       103
## 461653    1302470    Mazak, Kasey Ryne       102
## 1150254   3343788        Swift, Taylor       102
## 22081       60190     Andrisan, Rodrig       100
## 296401     831233 Harris, Lee Nicholas        97
```