Problem 1: How many actors are there in the database? How many movies?

For this problem, I found number of actors and actresses two ways: The first time I used id from role_type and used UNION to join all names from the name table. In the role_id, I called only for ids that were either a 1 or a 2 because those are the ids of female and male actors.

I'm really weary of cast_info because it takes too long to run and it has crashed my computer many times. While I tried to create a table for actors and I used ids from cast_info, My computer crashed and I had to restart it twice.

The first table I created combined the `cast_info role ids` with the`role_type ids`, and only used 1 or 2, (female and male actors) from each id column from those two tables. It took about 10 minutes to run, but I only had to run it once for it to be forever saved in my information.

the imdb_old table combined the ids from role_type and names (role ids only calling for 1 or 2, so male and female actors). This made it easier for me to count the names of actors, or how many actors there are in the database. There really was no need to do this tho, because the number of observations given in the global environment is the same as when I used a command to count the output from the table that I created.

{EDIT}!: I went back with the new lean, idx data and re-counted all my actors. My frist step was, I created a command with several inner joins that connected the cast_info2, name2, title2, and role_type tables together, in order to pull out actors of both sexes and count them. In order to make sure they were film actors, I selected kind_id='1' from the title2 table, because kind_id=1, calls only for movies. I connected columns from tabels that ahd equal columns, like id from title2 is equal to the movie_id column from cast_info2 table, and person_id from cast_info is equal to the id column from the name2 table.

After I created a temporary table with all this information, I then called for the counts of all the names from the temporary table. I could have done all of this in the same query, but have all my inner joins as a subquery, but this got kind of messy. I previously had created several tables and got a bunch of different results with my counts.

```r
setwd("E:/Fall 2015/141/Assignment 5")
library(RSQLite)

## Loading required package: DBI

imdb_new = dbConnect(SQLite(), dbname = "E:/Fall 2015/141/Assignment 5/lean_imdbpy_2010_idx.db")
```

```
 #create a temporary table of all actors from both sexes.
#I emphasis the kind_id=1 value to make sure they are film actors
#I connect tables with equal columns

dbGetQuery(imdb_new, "CREATE TEMPORARY TABLE count_actors AS
SELECT n.name AS count FROM title2 AS t
INNER JOIN cast_info2 AS c ON c.movie_id=t.id
INNER JOIN name2 AS n ON c.person_id=n.id
INNER JOIN role_type AS r ON c.role_id=r.id
WHERE kind_id=1 AND (r.role='actor' OR r.role='actress') ")



#in a separate query (thgis is just easier for me and saved me time) I
count all the names from my
#temporary table

dbGetQuery(imdb_new, "SELECT COUNT (distinct count) FROM count_actors")

##   COUNT (distinct count)
## 1                1030151

#1030151 distinct names
```

For the movies, I originally tried the `aka_titles` table, but the amount I got was very small, and something felt off with the number I got back, so I tried the `title` table and I got more results and this felt like better results to me. As someone who has used IMDB for over 10 years, getting back only 300k titles with `aka_titles` seemed very off to me.

I used the indexed database for this problem

```
setwd("E:/Fall 2015/141/Assignment 5")
library(RSQLite)

imdb_new = dbConnect(SQLite(), dbname = "E:/Fall 2015/141/Assignment
5/lean_imdbpy_2010_idx.db")




movi3es <- dbGetQuery(imdb_new, "SELECT COUNT(title) total_movies FROM
title2 WHERE kind_id='1'")

movi3es

##   total_movies
## 1       292918
```

2: What time period does the database cover?

I called for the MIN , the earliest year recorded in the `production_year` column from the title table, and I also called for the MAX or latest year recorded in the `production_year` column from the title table. I then looked up the film from 1974 and 2025 as a means of exploring the data further.

When using the `aka_title` column, I noticed that the MAX year was slightly different, I believe it was 2022 and not 2025.

I also thought the oldest film would be 'the great train robbery'.

Lean 8 tables reduced database was used for this problem.

```
setwd("E:/Fall 2015/141/Assignment 5")
library(RSQLite)

imdb = dbConnect(SQLite(), dbname = "E:/Fall 2015/141/Assignment
5/lean_imdbpy.db")

years_fun <- dbGetQuery(imdb, "SELECT MIN(production_year),
MAX(production_year) FROM title")

years_fun

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

#which movie is from 1874? and 2025?

dbGetQuery(imdb, "select production_year,id,title from title where
production_year='1874' ")

##   production_year      id             title
## 1            1874 3087258 Passage de Venus

#which movie is from 2025?

dbGetQuery(imdb, "select production_year,id,title from title where
production_year='2025';")

##   production_year      id         title
## 1            2025 3249359 StreetDance 4
```

Problem 3: What proportion of the actors are female? male?

For this I used piazza and a google search and after several attempts, managed to find an example that would work with the `cast_info`, specifically role_id column.

I did combine the `cast_info` role_ids with names, as a new table, but I had issues trying to get proportions of it out. I would like to in the future figure out a way to create a table, where at the same time too, create a column in the new table that I am creating/. I am sure there is a way to do this, but as of yet I have not figured that out.

The proportion I got back from the `cast_info` table:

There are 30% male actors and 16% female actors in the **indexed database**.

```
gender_proportion<-dbGetQuery(imdb_new, "SELECT role_id,
(COUNT(role_id)* 100.0 / (SELECT COUNT(*) FROM cast_info2)) AS Gender
FROM cast_info2
WHERE role_id ='1' OR role_id = '2'
GROUP BY role_id")

gender_proportion

##   role_id   Gender
## 1       1 29.89971
## 2       2 15.65358
```

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

Using the 8 tables reduced dataset, I changed this to the indexed dataset, I based this problem of the code that I found off some website that worked for problem 3 for this problem. The only real difference here is that there are MANY `WHERE` arguements. The only definitions that I know of, of any of these kind_id variables is that `kind_id='1' are movies, and kind_id='7' are tv-shows. Anything between 2-6 I am not sure of, because I couldn't find any column that defiend what all of these different 'kinds of media' are.

In the new dataset, there are only 4 different media types. The only one I am sure of, is '1' which is movies. In the new database, the breakdown is the following:

1. 81% movies
2. 11.8% of unknown media
3. 6.4% of unknown media 2
4. 0.8 % of unknown media form 3

```
setwd("E:/Fall 2015/141/Assignment 5")
imdb_new = dbConnect(SQLite(), dbname = "E:/Fall 2015/141/Assignment
5/lean_imdbpy_2010_idx.db")
library(RSQLite)

kind_test<-dbGetQuery(imdb_new, "SELECT kind_id,
(COUNT(kind_id)* 100.0 / (SELECT COUNT(*) FROM title2)) AS Proportion
FROM title2
WHERE kind_id ='1' OR kind_id = '2'
OR kind_id ='3' OR kind_id='4'
OR kind_id='5' OR kind_id='6' OR kind_id='7'
GROUP BY kind_id")

kind_test
```

```
##   kind_id Proportion
## 1       1 81.0056360
## 2       2 11.7878220
## 3       3  6.4040022
## 4       6  0.8025398
```