Querying a MySQL database

NOTE: The Rendered document was too long, so the professor reccomended that we shorten the document and to use limit.

This lab was originally created by Profs Ben Baumer and Jordan Crouser. I've edited it slightly.

SQL is a longstanding database querying language. It is a loosely-implemented standard. We will be using MySQL.

To facilitate our connection to the MySQL database server, we will need to install the `RMySQL` package.

```
# do NOT install RMySQL if you are on the RStudio Server
install.packages("RMySQL")
```

Goal: by the end of this lab, you will be able to write basic `SELECT` queries in SQL and retrieve the results into R.

Connecting to MySQL

The data we will be using is stored on a server in Bass Hall. It's called `scidb.smith.edu`. We can connect through the `dbConnect()` function provided by the `DBI` package (which is loaded automatically when you load `RMySQL`). You will also need the `RMySQL` package installed.

```
library(tidyverse)
library(RMySQL)
db <- dbConnect(
  MySQL(),
  host = "scidb.smith.edu",
  user = "sds192",
  password = "DSismfc@S",
  dbname = "imdb"
```

```
  )
knitr::opts_chunk$set(connection= 'db', max.print = 5)
```

This chunk of code will allow you to connect to `scidb`. Note that this creates a database connection object named `db`, which has the class `MySQLConnection`.

```
class(db)
```

```
[1] "MySQLConnection"
attr(,"package")
[1] "RMySQL"
```

Also, we set the `connection` parameter for all future chunks in this R Markdown file. Note also that the `max.print` argument sets the maximum number of results printed by each query.

Each of the following chunks makes use of the SQL engine functionality in `knitr`. You may want to read about this. Each of the following chunks is an `sql` chunk – as opposed to an `r` chunk!

To retrieve the results from a query in R, use the `dbGetQuery()` function from the `DBI` package (which is automatically loaded when you load `RMySQL`). Its first argument is a database connection object, and the second argument is an SQL query as a character vector.

Retrieving data

We want to be able to see which type of databases exist on the server. We can do this with SHOW DATABASES

```
dbGetQuery(conn = db,
           "SHOW DATABASES") |>
  head(5)
```

```
            Database
1           airlines
2           citibike
3                fec
4               imdb
5 information_schema
```

Note: If you had a problem on Friday and need to use dbGetQuery(conn=db, "SHOW DATABASES;) you will have to edit the code chunks below and above. Otherwise you can use the SQL chunks as printed.

You don't actually need the ; at the end of the quiery above for MySQL, but for other SQL dialects you do so it doesn't hurt.

We want to use the imdb databse and we have to tell our db connection that.

```
dbGetQuery(conn = db,"USE imdb;")
```

```
data frame with 0 columns and 0 rows
```

Its fine that there are 0 rows, this is just how we tell R which db we want to use.

Let's look at the tables available in imdb.

```
dbGetQuery(conn = db, "SHOW TABLES;")
```

```
    Tables_in_imdb
1         aka_name
2        aka_title
3        cast_info
4        char_name
5   comp_cast_type
6     company_name
7     company_type
8    complete_cast
9        info_type
10         keyword
11       kind_type
12       link_type
13 movie_companies
14      movie_info
15  movie_info_idx
16   movie_keyword
17      movie_link
18            name
19     person_info
20       role_type
21           title
```

See the kind_type table? That one shows what type of movie it is.

This query returns the list of kinds of "movies" stored in the IMDB. We are selecting everything with * from the kind_type table.

```
dbGetQuery(conn = db, "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
```

Of course, you will often want to store the result of your query as a data frame. This can be achieved by setting the `output.var` argument in the chunk. Here we retrieve the list of different types of information stored in the database, save it as a data frame in R, and show the first few rows.

```
## output.var="info_types" in this chunk

info_types<- dbGetQuery(conn = db, "SELECT * FROM info_type;")
```

We now have a `data.frame` called `info_types` in our environment.

```
# Note: this is an r chunk s we use R instead of SQL.

head(info_types)
```

```
  id         info
1  1     runtimes
2  2   color info
3  3       genres
4  4    languages
5  5 certificates
6  6    sound mix
```

That's all you need to know about how to get data from MySQL into R. The rest of this lab consists of practicing writing SQL queries. It may be useful to reference the full documentation for `SELECT` queries.

For example, let's say I wanted to find information on the wacky Bill Murray Movie Groundhog Day.

The titles are stored in the `title` field (i.e. column) in the `title` table. [Note: your professor is not responsible for naming these tables and fields!] Each row in the `title` table corresponds to a single movie, but of course, we need to restrict the rows we retrieve to only those where the `title` field equals `Groundhog Day`. The following query achieves this.

> Note: SQL does not require the `==` for testing equality, since you aren't ever changing the data.
>
> Note: You have to use `'` single quotes since you are working within a `"` double-quoted string.

In the chunk below we select every column from the title-table where the title-variable equals 'Groundhog Day'

```
dbGetQuery(conn = db,
           "SELECT *
           FROM title
           WHERE title= 'Groundhog Day'
           LIMIT 10;")
```

```
       id         title imdb_index kind_id production_year imdb_id
1   19605 Groundhog Day       <NA>       7            2014      NA
2   27895 Groundhog Day       <NA>       7            2008      NA
3  387076 Groundhog Day       <NA>       7            2011      NA
4  384111 Groundhog Day       <NA>       7            2016      NA
5  337084 Groundhog Day       <NA>       7            2016      NA
6  406720 Groundhog Day       <NA>       7            2008      NA
7  453412 Groundhog Day       <NA>       7            2013      NA
8  739375 Groundhog Day       <NA>       7            2003      NA
9  749553 Groundhog Day       <NA>       7            2005      NA
10 899995 Groundhog Day       <NA>       7            2015      NA
   phonetic_code episode_of_id season_nr episode_nr series_years
1          G6532         19602         1          2         <NA>
2          G6532         27822         1         48         <NA>
3          G6532        387052         1          2         <NA>
4          G6532        384073         1         11         <NA>
5          G6532        337080         1          5         <NA>
6          G6532        406715         1          6         <NA>
7          G6532        453388         1         39         <NA>
8          G6532        739088         7          1         <NA>
```

```
9          G6532        749522         3          4         <NA>
10         G6532        899984         4          1         <NA>
                             md5sum
1  a5e203197e1aa883f7884eb89e924aad
2  74ebd1bfceb83d4bdae480326d00b493
3  b8d621787a0ea75c76eff6b8a6a803c1
4  bcef9ee95ae2bb82eca4ae2341084054
5  fcaddb8be4ab7b9c5ca9f27bb325d869
6  07d71924ad7cb3e2cbc26b36c7b947ec
7  a5319edde61625b2a812edd32e552821
8  419502b27afaa77a170c2ed0c07be5da
9  c25e2aa6984230afe9163f81f23b04fb
10 033a2f4e4fbe2d00068ef81c2979a744
```

That retrieved a lot of movies! Let's see if we can refine our query. First, movies (as opposed to TV episodes, etc.) have the `kind_id` value of `1`.

```
dbGetQuery(conn = db, "SELECT *
           FROM title
           WHERE title = 'Groundhog Day'
           AND kind_id = 1;")
```

```
      id         title imdb_index kind_id production_year imdb_id
1 3664274 Groundhog Day       <NA>       1            1993      NA
  phonetic_code episode_of_id season_nr episode_nr series_years
1         G6532            NA        NA         NA         <NA>
                            md5sum
1 2f0a563d0b0a1f57a19385de5a8770e2
```

Now we have the result that I want.

Imagine that I didn't know the full title of the movie I could soften my query by searching for the phrase `Groundhog` within the title. We can do this using the `LIKE` function along with some wildcards (`%` in SQL).

```
dbGetQuery(conn = db, "SELECT *
           FROM title
           WHERE title LIKE '%Groundhog%'
           AND kind_id = 1;")
```

```
      id           title imdb_index kind_id production_year imdb_id
```

```
1 3664274   Groundhog Day       <NA>       1            1993      NA
2 3664277      Groundhogs       <NA>       1            2015      NA
3 3664273       Groundhog       <NA>       1            2017      NA
4 3664276 Groundhog's Day       <NA>       1            2016      NA
  phonetic_code episode_of_id season_nr episode_nr series_years
1         G6532            NA        NA         NA         <NA>
2         G6532            NA        NA         NA         <NA>
3         G6532            NA        NA         NA         <NA>
4         G6532            NA        NA         NA         <NA>
                           md5sum
1 2f0a563d0b0a1f57a19385de5a8770e2
2 7b71cb8ae79de1171a71f95d2e50afd6
3 5e7183dbeb6c28fb6445c4013b2bd5d0
4 c45dd6456b9787e5f71144d6c3a2295e
```

Pretend I'm still not sure which of the above four movies is the real Groundhog Day movie I'm interested in, but I'm sure its the first movie that came out. I could put them in order with the code below.

```
dbGetQuery(conn = db, "SELECT *
           FROM title
           WHERE title LIKE '%Groundhog%'
           AND kind_id = 1
           ORDER BY production_year;")
```

```
      id           title imdb_index kind_id production_year imdb_id
1 3664274   Groundhog Day       <NA>       1            1993      NA
2 3664277      Groundhogs       <NA>       1            2015      NA
3 3664276 Groundhog's Day       <NA>       1            2016      NA
4 3664273       Groundhog       <NA>       1            2017      NA
  phonetic_code episode_of_id season_nr episode_nr series_years
1         G6532            NA        NA         NA         <NA>
2         G6532            NA        NA         NA         <NA>
3         G6532            NA        NA         NA         <NA>
4         G6532            NA        NA         NA         <NA>
                           md5sum
1 2f0a563d0b0a1f57a19385de5a8770e2
2 7b71cb8ae79de1171a71f95d2e50afd6
3 c45dd6456b9787e5f71144d6c3a2295e
4 5e7183dbeb6c28fb6445c4013b2bd5d0
```

Finally I can select just the three columns I'm interested in. Also notice that I am renaming the title table as t. So I select the columns t.title and t.production_year. This is called creating an alias. The convention is table.variable. This will be useful when joining tables.

```
dbGetQuery(conn = db, "SELECT t.id, t.title, t.production_year
           FROM title AS t
           WHERE title LIKE '%Groundhog%'
           AND t.kind_id = 1
           ORDER BY t.production_year;")
```

```
       id           title production_year
1 3664274   Groundhog Day            1993
2 3664277      Groundhogs            2015
3 3664276 Groundhog's Day            2016
4 3664273       Groundhog            2017
```

Its the first Groundhog day that came out in 1993 with ID 3664274.

Exercise:

Find the original Ghostbusters in the `title` table.

```
dbGetQuery(conn = db, "SELECT *
           FROM title
           WHERE title LIKE '%Ghostbusters%'
           AND kind_id = 1
           LIMIT 10;")
```

```
       id                         title imdb_index kind_id production_year
1  3501764          Detroit GhostBusters       <NA>       1            2013
2  3644559 Ghostbusters Ecto-1 Jailbreak       <NA>       1            2016
3  3644563  Ghostbusters Italia Fan Film       <NA>       1            2017
4  3644560               Ghostbusters II       <NA>       1            1989
5  3644567  Ghostbusters vs. Mythbusters       <NA>       1            2013
6  3644565              Ghostbusters SLC       <NA>       1            2010
7  3644554                  Ghostbusters       <NA>       1            2016
8  3644564           Ghostbusters Return       <NA>       1            2010
9  3644550                  Ghostbusters       <NA>       1            1984
10 3644558                Ghostbusters 3       <NA>       1            2012
```

```
   imdb_id phonetic_code episode_of_id season_nr episode_nr series_years
1       NA         D3632            NA        NA         NA         <NA>
2       NA         G2312            NA        NA         NA         <NA>
3       NA         G2312            NA        NA         NA         <NA>
4       NA         G2312            NA        NA         NA         <NA>
5       NA         G2312            NA        NA         NA         <NA>
6       NA         G2312            NA        NA         NA         <NA>
7       NA         G2312            NA        NA         NA         <NA>
8       NA         G2312            NA        NA         NA         <NA>
9       NA         G2312            NA        NA         NA         <NA>
10      NA         G2312            NA        NA         NA         <NA>
                             md5sum
1  cb3d50c572ba5539b93554629d9bc361
2  e13b6ccdd089c7503918bd49036a040b
3  ae21c91c8c217fc0a6adabc4ca60a9a5
4  b111b3d12d8e96ce76a13f3d2faf028b
5  b7f4191d55c3ff35466ffa4ddaa7ee34
6  39919bfc3094bd5e125c8c1958f67c9b
7  9dc8686712d01fc31d597c8aa1893346
8  a483b2cab445e9301ea6187ec1c2aa92
9  595c84db94e698e6002097cd7c2b7849
10 90a52baa0f97758adacbf69ce4c4d34c
```

Now lets consider the name table

```
dbGetQuery(conn= db, "DESCRIBE name;")
```

```
          Field        Type Null Key Default          Extra
1            id         int   NO PRI    <NA> auto_increment
2          name        text   NO MUL    <NA>
3    imdb_index varchar(12)  YES        <NA>
4       imdb_id         int  YES MUL    <NA>
5        gender  varchar(1)  YES        <NA>
6 name_pcode_cf  varchar(5)  YES MUL    <NA>
7 name_pcode_nf  varchar(5)  YES MUL    <NA>
8 surname_pcode  varchar(5)  YES MUL    <NA>
9        md5sum varchar(32)  YES MUL    <NA>
```

Exercise:

Find Andie MacDowell's `id` in the `name` table.

Note: that names are listed last name first and seperated by a comma (eg Murray, Bill)

```
dbGetQuery(conn= db,"SELECT id
           FROM name
           WHERE name LIKE 'MacDowell, Andie%';")
```

```
      id
1 3479179
```

Joining tables

In the IMDB, the `title` table contains information about movies, the `name` table contains the names of people, the `char_name` table contains information about the names of characters, and the `cast_info` table contains information about which people played which roles in which movies. Linking the tables together is essential in order to extract information from the database.

Since we already know that the ID of *Groundhog Day* is `3664274`, we can use that to find all of the cast assignments.

```
dbGetQuery(conn= db, "SELECT *
           FROM cast_info
           WHERE movie_id = 3664274
           LIMIT 10;")
```

```
       id person_id movie_id person_role_id         note nr_order role_id
1  118176     17392  3664274            376         <NA>       43       1
2 1764292    232622  3664274         352541 (uncredited)       NA       1
3 2746369    354870  3664274         191351         <NA>       13       1
4 3167452    400710  3664274         140101         <NA>       34       1
5 4584838    577860  3664274             98 (uncredited)       NA       1
6 4725830    596320  3664274         264682 (uncredited)       NA       1
7 5044054    638176  3664274          42869         <NA>        5       1
8 5100344    646080  3664274         880820         <NA>       22       1
9 5106264    647033  3664274           1913         <NA>        8       1
10 5395988   685478  3664274           5433         <NA>        3       1
```