DATA 607 Assignment 2

Magnus Skonberg

9/2/2020

OVERVIEW

The purpose of this assignment was to collect data, store it in a relational (SQL) database, and then import this data into an R dataframe to analyze it.

I polled (6) immediate family members and/or significant others regarding (5) movies that had come out in the past year, asked for their rating, from 1 to 5, and stored their response or a -1 (if they hadn't seen the movie). I processed these -1s in SQL by replacing the -1 with a null value.

To store these results and build a relational database in SQL, I used (3) separate .csv files - movies, raters, and ratings. These tables are all available on github: https://github.com/Magnus-PS/CUNY-SPS-DATA-607/tree/Assignment-2.

STORING DATA IN A RELATIONAL DATABASE

The corresponding SQL code used to create and populate tables from corresponding .csv is provided below:

```
/* CUNY SPS - DATA 607 - Assignment 2 */
/* Student: Magnus Skonberg */
/* Date: September 2nd 2020 */

/* Purpose: to import data from a .csv and populate tables in mySQL to create a relational database. */

DROP TABLE IF EXISTS rater;
DROP TABLE IF EXISTS movie;
DROP TABLE IF EXISTS rating;

CREATE TABLE rater(
raterID int NOT NULL PRIMARY KEY,
raterName VARCHAR(10) NOT NULL UNIQUE
  );

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/rater.csv'
INTO TABLE rater
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(raterID, raterName);

CREATE TABLE movie
(
movieID int NOT NULL PRIMARY KEY,
```

```
movieTitle varchar(30) NOT NULL UNIQUE
);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movie.csv'
INTO TABLE movie
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(movieID, movieTitle);

CREATE TABLE rating(
movieID int NOT NULL REFERENCES movie(movieID),
raterId int NOT NULL REFERENCES rater(raterID),
rating int NULL
  );

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/rating.csv'
INTO TABLE rating
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(movieID, raterID, @rating)
SET rating = nullif(@rating, -1);

select * from movie;
select * from rater;
select * from rating;
```

Once the relational database had been built out (using SQL), I set out to establish a connection between RStudio and MySQL to import this data.

CONNECT TO MYSQL

Write code in R markdown to import DB data (from SQL) for analysis.
Missing data was handled during SQL load.

```
library(RCurl);
library(DBI);
library(RMySQL);
library(ggplot2);

movieDB <- dbConnect(MySQL(), user='root', password='pass123', dbname='movie_ratings', host='localhost')
```

Once the connection had been established, I read from the SQL database.

Note: this step took a good amount of time / effort due to a bug in the 8.0.21 version of MySQL for Windows10. To get around this issue, I uninstalled the 8.0.21 version and installed the 8.0.20 version of MySQL and altered the password encryption to allow a connection between MySQL and RStudio.

```
dbListTables(movieDB);
```

```
## [1] "movie"  "rater"  "rating"
```

```
new_movie <- dbReadTable(movieDB, "movie")

new_rater <- dbReadTable(movieDB, "rater")

new_rating <- dbReadTable(movieDB, "rating")

new_movie
```

```
##   movieID                    movieTitle
## 1       4                      Capone\r
## 2       2                   Eurovision\r
## 3       5 Impractical Jokers: the Movie
## 4       1                 Project Power\r
## 5       3              The Wrong Missy\r
```

```
new_rater
```

```
##   raterID  raterName
## 1       1   Carl S\r
## 2       4  Emily C\r
## 3       6     Kian D
## 4       2  Laine S\r
## 5       5   Ryan Q\r
## 6       3 Stefan S\r
```

```
new_rating
```

```
##    movieID raterId rating
## 1        1       1      4
## 2        1       2      3
## 3        1       3      3
## 4        1       4      5
## 5        1       5     NA
## 6        2       1      2
## 7        2       2      5
## 8        2       3      4
## 9        2       4      3
## 10       2       5     NA
## 11       3       1      4
## 12       3       2      5
## 13       3       3      4
## 14       3       4      3
## 15       3       5      5
## 16       4       1      4
## 17       4       2      5
## 18       4       3      5
## 19       4       4     NA
## 20       4       5      3
## 21       5       1      5
## 22       5       2      5
## 23       5       3      5
## 24       5       4      2
## 25       5       5      4
## 26       6       1      4
```

```
## 27       6       2       4
## 28       6       3       4
## 29       6       4      NA
## 30       6       5      NA
```

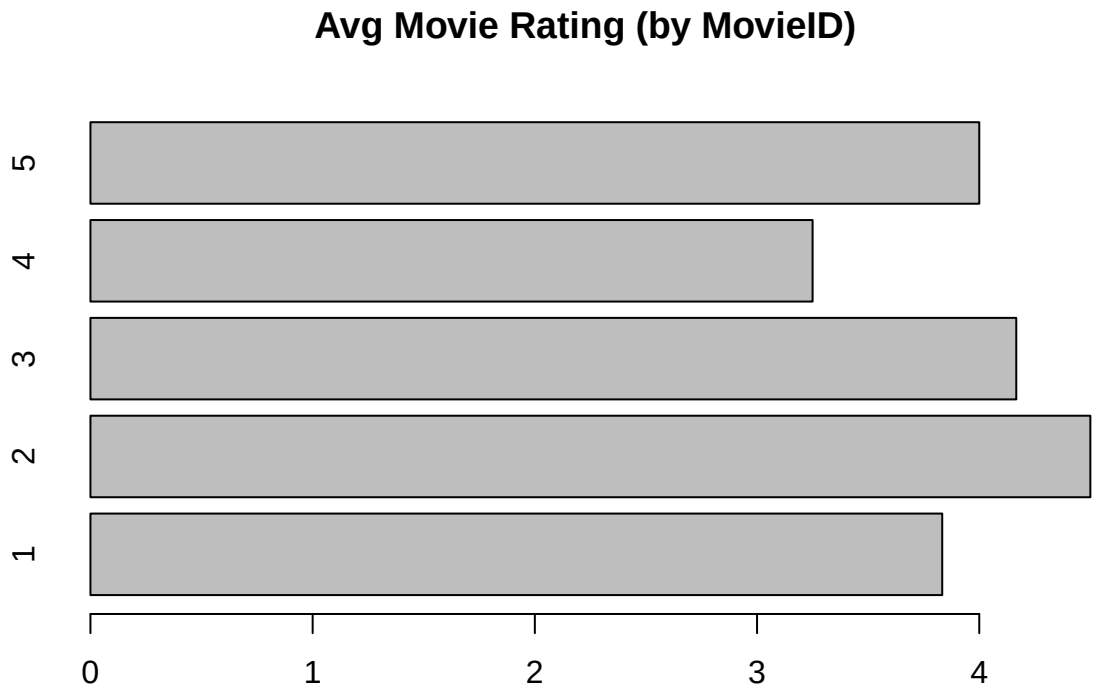
INCLUDE A QUALITY GRAPHIC OR TABLE

To display the Avg Movie Rating (by MovieID), I deemed it necessary to create a table of the average rating plot per movie ID. To find this avg_rating number, the sum of (non-null) ratings per movieID was taken over the number of the number of non-null ratings.

```
movie <- 1:5
avg_rating <- c(23/6, 27/6, 25/6, 13/4, 12/3)

avg <- as.table(setNames(avg_rating, movie))

barplot(avg, main="Avg Movie Rating (by MovieID)", horiz=TRUE,
  names.arg=c("1", "2", "3", "4", "5"))
```



ANALYSIS & CONCLUSION

Based on the above barplot, we can see that all movies had an average rating of 3 or higher with movie #4 (Capone) having the lowest avg movie rating and that movie #2 (Eurovision) having the highest avg movie rating. Thus, if I were to recommend movies to this particular subset of movie viewers, a comedy similar to 'Eurovision' would be a better recommendation than a drama bio-flick like 'Capone'.