# Assignment02

Ted Kim

2022-09-09

### Team Member

Seung Min Song

## OVERVIEW

Choose six recent popular movies and ask at least five friends and family members who know to rate each movie they have seen from 1 to 5. Get the results, save them to the My SQL database, and load the information in the SQL database into the R data frame. Assuming that there is no existing database, I created a database and a table. The query generating the database and table was inserted as hardcoding in the markdown. The movie-related data was downloaded from themoviedb.org as a json file via api. Json files are registered on GitHub. It can be implemented simply by using the overwrite property of the dbWriteTable() function, but since it is not possible to create a relationship between tables, the append property will be used. Therefore, before each table is stored, duplicate data is discarded by comparing it with the existing table and json/Google sheet.

### CONNECT MySQL

Create a connection object to MySQL database.

```
con = dbConnect(RMySQL::MySQL(),
                user='root',
                password='',
                host='localhost')
```

### CREATE DATABASE

In general, databases and tables are often already created. However, if a table is not created, we must create a table and put data in it when the program runs. Create database *movies* if not exists, and use database *movies*

```r
res <- dbSendQuery(con, 'create database if not exists movies;')

res <- dbSendQuery(con, 'use movies;')
```

**CREATE TABLES**

Create tables *genres, movies, movies_genres* if not exists

```r
res <- dbSendQuery(con, 'create table if not exists genres (
                            id int primary key,
                            description varchar(40)
                        ) engine=innodb;')

res <- dbSendQuery(con, 'create table if not exists movies (
                            movie_id int auto_increment primary key,
                            adult bool,
                            backdrop_path varchar(255),
                            original_language varchar(10),
                            original_title varchar(255),
                            overview varchar(1000),
                            popularity double(8,3),
                            poster_path varchar(255),
                            release_date date,
                            title varchar(255),
                            video bool,
                            vote_average double(8,3),
                            vote_count int
                        ) engine=innodb;')

res <- dbSendQuery(con, 'create table if not exists movies_genres (
                            id int auto_increment primary key,
                            movie_id int,
                            genre_id int,
                            foreign key (movie_id) references movies (movie_id),
                            foreign key (genre_id) references genres (id)
                        ) engine=innodb;')
```

```
res <- dbSendQuery(con, 'create table if not exists survey_result (
                            id int auto_increment primary key,
                            survey_id int,
                            movie_id int,
                            email_address varchar(255),
                            rate int,
                            registered datetime,
                            foreign key (movie_id) references movies (movie_id)
                        ) engine=innodb;')
```

Displays the tables available in the database

```
## [1] "genres"        "movies"        "movies_genres" "survey_result"
```

## LOAD GENRES DATA

Stores genre data imported from TMDB into the table on SQL Server. Since the same data may exist in the table on SQL Server, only subsets of the two data that do not exist in the table on SQL Server are appended.

**Get genres data from SQL server**

```
dfGenres <- fetch(dbSendQuery(con, 'select * from genres'), )
dbClearResult(dbListResults(con)[[1]])
```

[1] TRUE

**Get genres.json from github to load**

```
file <- 'https://raw.githubusercontent.com/blacksmilez/DATA607/main/Assignment02/json/genres.json'
jsonGenres <- fromJSON(file)
```

**Retrieve all rows on data frame *dfGenres, jsonGenres* to verify**

```
print(dfGenres[order(dfGenres$id),], row.names = FALSE, right = FALSE)
```

```
##  id    description
##     12 Adventure
##     14 Fantasy
##     16 Animation
##     18 Drama
##     27 Horror
##     28 Action
##     35 Comedy
##     36 History
##     37 Western
##     53 Thriller
##     80 Crime
##     99 Documentary
##    878 Science Fiction
##   9648 Mystery
##  10402 Music
##  10749 Romance
##  10751 Family
##  10752 War
##  10770 TV Movie
```

```
print(jsonGenres[order(jsonGenres$id),], row.names = FALSE, right = FALSE)
```

```
##  id    description
##     12 Adventure
##     14 Fantasy
##     16 Animation
##     18 Drama
##     27 Horror
##     28 Action
##     35 Comedy
##     36 History
##     37 Western
##     53 Thriller
##     80 Crime
##     99 Documentary
##    878 Science Fiction
##   9648 Mystery
##  10402 Music
##  10749 Romance
##  10751 Family
##  10752 War
##  10770 TV Movie
```

**Get Subsets of the two data that do not exist in the table on SQL Server**

```
## [1] id          description
## <0 rows> (or 0-length row.names)
```

**Append subsets into the table on SQL Server**

```
dbWriteTable(con, 'genres', dfSubsets[, c('id', 'description')], row.names=FALSE, append=TRUE)
```

```
## [1] TRUE
```

## LOAD MOVIES DATA

Stores movies data imported from TMDB into the table on SQL Server. Since the same data may exist in the table on SQL Server, only subsets of the two data that do not exist in the table on SQL Server are appended.

**Get movies data from SQL server**

```
dfMovies <- fetch(dbSendQuery(con, 'select * from movies'), )
dbClearResult(dbListResults(con)[[1]])
```

[1] TRUE

**Get movies.json from github to load**

```
file <- 'https://raw.githubusercontent.com/blacksmilez/DATA607/main/Assignment02/json/movies.json'
jsonMovies <- fromJSON(file)
```

**Retrieve all rows on data frame *dfMovies*, *jsonMovies* to verify**

```
print(dfMovies[order(dfMovies$movie_id), c('movie_id', 'title', 'release_date')],
      row.names = FALSE, right = FALSE)
```

```
##  movie_id title                  release_date
##  361743   Top Gun: Maverick      2022-05-24
##  507086   Jurassic World Dominion 2022-06-01
##  539681   DC League of Super-Pets 2022-07-27
##  629176   Samaritan              2022-08-25
##  634649   Spider-Man: No Way Home 2021-12-15
##  755566   Day Shift              2022-08-10
```

```
print(jsonMovies[order(jsonMovies$movie_id), c('movie_id', 'title', 'release_date')],
      row.names = FALSE, right = FALSE)
```

```
##  movie_id title                  release_date
##  361743   Top Gun: Maverick      2022-05-24
##  507086   Jurassic World Dominion 2022-06-01
##   539681   DC League of Super-Pets 2022-07-27
##   629176   Samaritan               2022-08-25
##   634649   Spider-Man: No Way Home 2021-12-15
##   755566   Day Shift               2022-08-10
```

**Get Subsets of the two data that do not exist in the table on SQL Server**

```
## [1] movie_id     title        release_date
## <0 rows> (or 0-length row.names)
```

**Append subsets into the table on SQL Server**

```
dbWriteTable(con, 'movies',
             dfSubsets[, c('movie_id', 'adult', 'backdrop_path', 'original_language',
                          'original_title', 'overview', 'popularity', 'title',
                          'poster_path', 'release_date', 'video')],
             row.names=FALSE, append=TRUE)
```

```
## [1] TRUE
```

## LOAD MOVIES-GENRES DATA

Stores movies_genres data imported from TMDB into the table on SQL Server. Since the same data may exist in the table on SQL Server, only subsets of the two data that do not exist in the table on SQL Server are appended.

**Get movies_genres data from SQL server**

```
dfMoviesGenres <- fetch(dbSendQuery(con, 'select * from movies_genres'), )
dbClearResult(dbListResults(con)[[1]])
```

[1] TRUE

**Get movies_genres.json from github to load**

```
file <- 'https://raw.githubusercontent.com/blacksmilez/DATA607/main/Assignment02/json/movies_genres.jso
jsonMoviesGenres <- fromJSON(file)
```

**Retrieve all rows on data frame *dfMoviesGenres*, *jsonMoviesGenres* to verify**

```
print(dfMoviesGenres[order(dfMoviesGenres$movie_id, dfMoviesGenres$genre_id),],
      row.names = FALSE, right = FALSE)
```

```
##   id movie_id genre_id
##   13 361743      18
## 12 361743      28
##  9 507086      12
## 10 507086      28
## 11 507086     878
##  4 539681      16
##  5 539681      28
##  8 539681      35
##  7 539681     878
##  6 539681   10751
##  2 629176      18
## 22 629176      18
##  1 629176      28
## 21 629176      28
##  3 629176     878
## 23 629176     878
## 19 634649      12
## 18 634649      28
## 20 634649     878
## 15 755566      14
## 16 755566      27
## 14 755566      28
## 17 755566      35
```

```
print(jsonMoviesGenres[order(jsonMoviesGenres$movie_id, jsonMoviesGenres$genre_id),],
      row.names = FALSE, right = FALSE)
```

```
## movie_id genre_id
##  361743     18
##  361743     28
##  453395     12
##  453395     14
##  453395     28
##  507086     12
##  507086     28
##  507086    878
##  539681     16
##  539681     28
##   539681     35
##   539681    878
##   539681  10751
##   616037     12
##   616037     14
##   616037     28
##   629176     18
##   629176     18
##   629176     28
##   629176     28
##   629176    878
##   629176    878
##   634649     12
##   634649     28
##   634649    878
##   755566     14
##   755566     27
##   755566     28
##   755566     35
##   766507     28
##   766507     53
##   848123     28
##   848123     53
##   927341     27
##   927341     28
##   927341     53
##   951368     28
##   951368     53
##   951368  10770
##   997120     28
##   997120     53
##   997120   9648
##  1006851     28
##  1006851     35
##  1006851    878
##  1008779     28
```

#### Get Subsets of the two data that do not exist in the table on SQL Server

```
## [1] movie_id genre_id
## <0 rows> (or 0-length row.names)
```

#### Append subsets into the table on SQL Server

```
dbWriteTable(con, 'movies_genres',
             dfSubsets[, c('movie_id', 'genre_id')],
             row.names=FALSE, append=TRUE)
```

```
## [1] TRUE
```

※ The following error occured when running "dbWriteTable()" for the first time:
*"ERROR: Loading local data is disabled - this must be enabled on both the client and server sides"*
error occurs while copying data frames to database tables using dbWriteTable(), it is handled as follows:

```
# 1. open mysql terminal
# 2. check the local_infile
#    mysql> show global variables like 'local_infile'
#    +---------------+-------+
#    | Variable_name | Value |
#    +---------------+-------+
#    | local_infile  |  OFF  |
#    +---------------+-------+
#    (this means local_infile is disable)
# 3. put set command
#    mysql> set global local_infile=true;
#    mysql> exit
```

## LOAD SURVEY DATA

Stores survey data imported from Google Sheet into the table on SQL Server. Since the same data may exist in the table on SQL Server, only subsets of the two data that do not exist in the table on SQL Server are appended.

#### Get survey result data from SQL server

Obtain survey data from SQL Server. There is no record set returned because there is no data at the time of initial execution.

```r
dfSurveyResults <- fetch(
                    dbSendQuery(con, 'select * from survey_result where survey_id = 1'),)
dbClearResult(dbListResults(con)[[1]])
```

```
[1] TRUE
```

```r
print(dfSurveyResults[order(dfSurveyResults$movie_id),], row.names = FALSE, right = FALSE)
```

```
id survey_id movie_id email_address rate registered
1 1 361743 blacksmilez@gmail.com 4 2022-09-11 12:19:59 2 1 361743 negativetae@gmail.com 5 2022-09-11
12:21:38 3 1 361743 nury95@hotmail.com 5 2022-09-11 12:47:01 4 1 361743 s88724@gmail.com 5 2022-09-
11 13:21:36 5 1 507086 blacksmilez@gmail.com 5 2022-09-11 12:19:59 6 1 507086 negativetae@gmail.com 1
2022-09-11 12:21:38 7 1 507086 nury95@hotmail.com 5 2022-09-11 12:47:01 8 1 507086 s88724@gmail.com
2 2022-09-11 13:21:36 14 1 539681 blacksmilez@gmail.com 3 2022-09-11 12:19:59 15 1 539681 negativetae@
gmail.com 1 2022-09-11 12:21:38 16 1 539681 nury95@hotmail.com 5 2022-09-11 12:47:01 17 1 539681 delight_
32@hotmail.com 4 2022-09-11 13:13:58 18 1 539681 s88724@gmail.com 3 2022-09-11 13:21:36 19 1 629176
blacksmilez@gmail.com 2 2022-09-11 12:19:59 20 1 629176 negativetae@gmail.com 3 2022-09-11 12:21:38 21
1 629176 nury95@hotmail.com 5 2022-09-11 12:47:01 22 1 629176 delight_32@hotmail.com 4 2022-09-11
13:13:58 23 1 629176 s88724@gmail.com 4 2022-09-11 13:21:36 9 1 634649 blacksmilez@gmail.com 4 2022-
09-11 12:19:59 10 1 634649 negativetae@gmail.com 2 2022-09-11 12:21:38 11 1 634649 nury95@hotmail.com
3 2022-09-11 12:47:01 12 1 634649 delight_32@hotmail.com 3 2022-09-11 13:13:58 13 1 634649 s88724@
gmail.com 4 2022-09-11 13:21:36 24 1 755566 negativetae@gmail.com 4 2022-09-11 12:21:38 25 1 755566
nury95@hotmail.com 3 2022-09-11 12:47:01 26 1 755566 delight_32@hotmail.com 4 2022-09-11 13:13:58 27
1 755566 s88724@gmail.com 3 2022-09-11 13:21:36
```

**Get survey result from google sheet to load**

The survey was conducted with Google Forms that can be easily used.

hyperlink: https://docs.google.com/forms/d/e/1FAIpQLSeL4Ymj956wxJ9rMH-ie-XHgmg6P-d25iHjvxAyNmKc7QIvIg/view

It brings up the Google sheet where the data input through Google Form is stored.

```r
gs4_deauth()
file <- 'https://docs.google.com/spreadsheets/d/1n8U9AbOSKMI871oHoycPK-WXcKkC3_VmfdSx742hbTo/edit?usp=sl
df <- as.data.frame(read_sheet(file))
```

**Get survey result from google sheet to load**

The ID of the movie used for the survey is included in the array. A separate metric table should be created, but this time it will be omitted.

```
movies_id <- c(361743, 507086, 634649, 539681, 629176, 755566)
```

Creates an empty data frame for storing subsets.

```
dfSurveySubsets <- data.frame(matrix(ncol=5, nrow=0))
colnames(dfSurveySubsets) <- c('survey_id', 'movie_id', 'email_address', 'rate', 'registered')
```

Only new survey data that does not exist in the existing survey table is selected from the Google sheet.

```
count <- 1
for(id in movies_id) {
  dftmp <- df[, c('survey_id', paste0('movie', count), 'email_address', 'registered')]
  names(dftmp)[names(dftmp) == paste0('movie', count)] <- 'rate'
  dftmp['movie_id'] <- id
  dfSubsets <- subset(dftmp, !(movie_id %in% dfSurveyResults$movie_id
                              &&email_address %in% dfSurveyResults$email_address))
  dfSurveySubsets <- rbind(dfSurveySubsets, dfSubsets)
  count = count + 1
}
print(dfSurveySubsets)
```

[1] survey_id movie_id email_address rate registered
<0 rows> (or 0-length row.names)

**Append subsets into the table on SQL Server**

Only non-duplicated data is stored in the SQL Server table.

```
dbWriteTable(con, 'survey_result',
             dfSurveySubsets[dfSurveySubsets$rate > 0,],
             row.names=FALSE, append=TRUE)
```

```
## [1] TRUE
```

**Re-Get survey result data from SQL server**

The survey data is retrieved from the SQL server again.

```
dfSurveyResults <- fetch(dbSendQuery(con, 'select * from survey_result where survey_id = 1'),)
dbClearResult(dbListResults(con)[[1]])
```

[1] TRUE

Make the column names the same for join between two data frames.

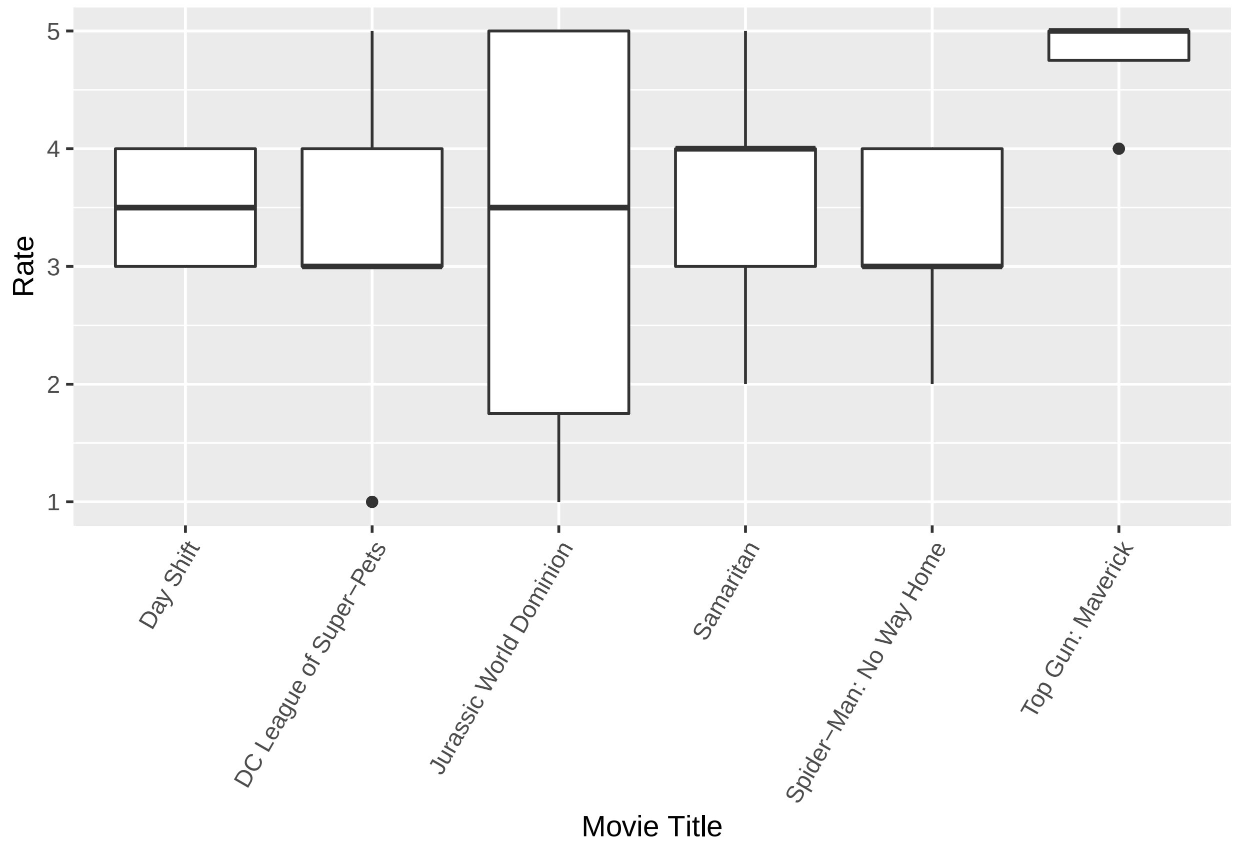
```
names(jsonMovies)[names(jsonMovies) == 'id'] <- 'movie_id'
```

Merge the two data tables and remove unnecessary columns.

```
dfResults <- merge(dfSurveyResults, jsonMovies, by = 'movie_id')
dftmp <- subset(dfResults, select = -c(registered, adult, backdrop_path, original_language,
                                        overview, original_title, poster_path, release_date,
                                        video, id, survey_id))
```

**Graphs drawn without calibration of missing data**

If the graph is drawn without calibration of missing data as follows.



## Missing Data (1)

**For calibration, the missing values can be filled with *mean* values.**

```
colnames(dftmp)
```

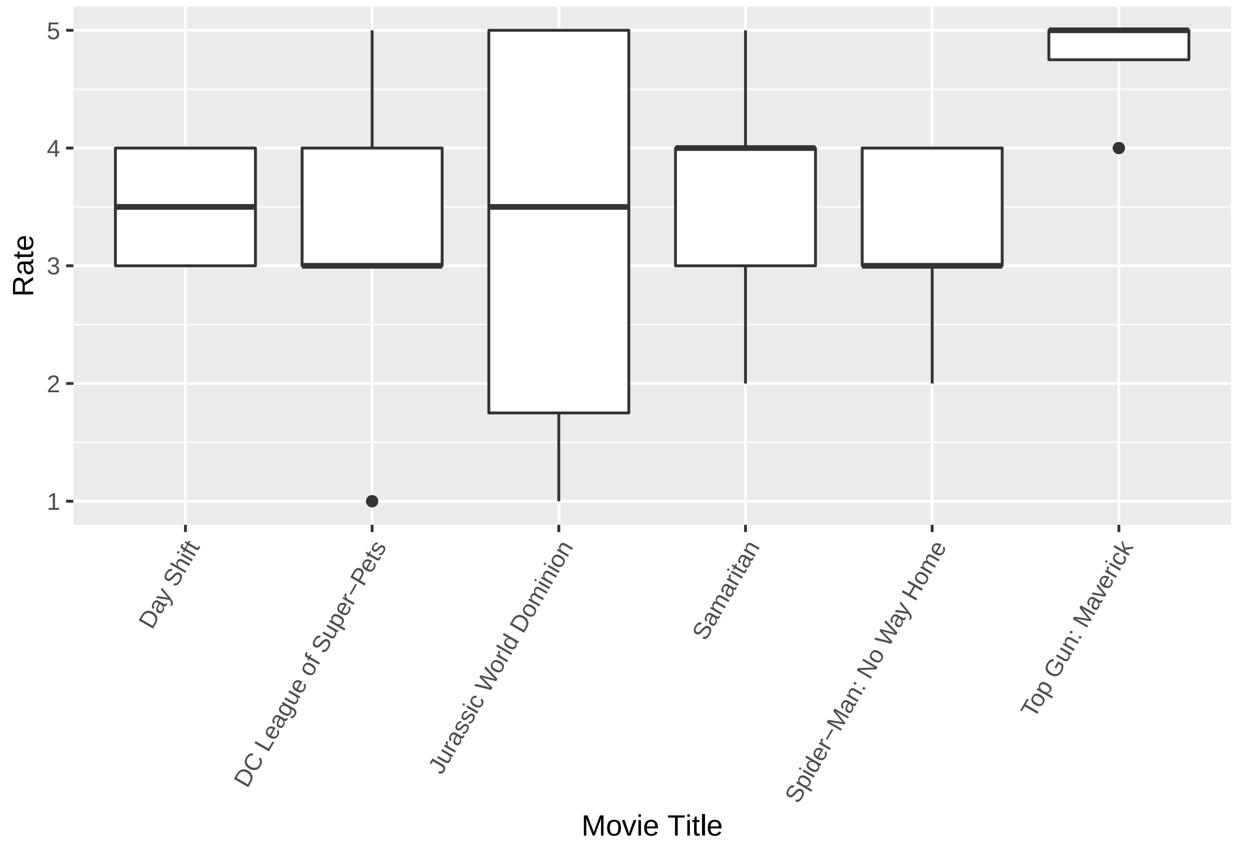
```
## [1] "movie_id"      "email_address" "rate"          "popularity"
## [5] "title"
```

```
dfmean <- dftmp %>%
        group_by(movie_id) %>%
        mutate(mean = mean(rate))

dfmean$rate <- ifelse(dfmean$rate == 0, dfmean$mean, dfmean$rate)

print(dfmean, n=100)
```

```
## # A tibble: 27 x 6
## # Groups:   movie_id [6]
##    movie_id email_address           rate popularity title                    mean
##       <int> <chr>                  <int>      <dbl> <chr>                   <dbl>
##  1   361743 blacksmilez@gmail.com      4      2030. Top Gun: Maverick        4.75
##  2   361743 negativetae@gmail.com      5      2030. Top Gun: Maverick        4.75
##  3   361743 nury95@hotmail.com         5      2030. Top Gun: Maverick        4.75
##  4   361743 s88724@gmail.com           5      2030. Top Gun: Maverick        4.75
##  5   507086 blacksmilez@gmail.com      5      2084. Jurassic World Domini~   3.25
##  6   507086 negativetae@gmail.com      1      2084. Jurassic World Domini~   3.25
##  7   507086 nury95@hotmail.com         5      2084. Jurassic World Domini~   3.25
##  8   507086 s88724@gmail.com           2      2084. Jurassic World Domini~   3.25
##  9   539681 blacksmilez@gmail.com      3      2738. DC League of Super-Pe~   3.2
## 10   539681 negativetae@gmail.com      1      2738. DC League of Super-Pe~   3.2
## 11   539681 nury95@hotmail.com         5      2738. DC League of Super-Pe~   3.2
## 12   539681 delight_32@hotmail.com     4      2738. DC League of Super-Pe~   3.2
## 13   539681 s88724@gmail.com           3      2738. DC League of Super-Pe~   3.2
## 14   629176 blacksmilez@gmail.com      2      5115. Samaritan                3.6
## 15   629176 negativetae@gmail.com      3      5115. Samaritan                3.6
## 16   629176 nury95@hotmail.com         5      5115. Samaritan                3.6
## 17   629176 delight_32@hotmail.com     4      5115. Samaritan                3.6
## 18   629176 s88724@gmail.com           4      5115. Samaritan                3.6
## 19   634649 blacksmilez@gmail.com      4      1142. Spider-Man: No Way Ho~   3.2
## 20   634649 negativetae@gmail.com      2      1142. Spider-Man: No Way Ho~   3.2
## 21   634649 nury95@hotmail.com         3      1142. Spider-Man: No Way Ho~   3.2
## 22   634649 delight_32@hotmail.com     3      1142. Spider-Man: No Way Ho~   3.2
## 23   634649 s88724@gmail.com           4      1142. Spider-Man: No Way Ho~   3.2
## 24   755566 negativetae@gmail.com      4      1403. Day Shift                3.5
## 25    755566 nury95@hotmail.com         3       1403. Day Shift               3.5
## 26    755566 delight_32@hotmail.com     4       1403. Day Shift               3.5
## 27    755566 s88724@gmail.com           3       1403. Day Shift               3.5
```

**Graph of missing values filled with *mean* values**

```
ggplot(dfmean, aes(x=title, y=rate)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1.0 )) +
  labs(x='Movie Title', y='Rate')
```



## Missing Data (2)

**For calibration, the missing values can be filled with *median* values.**

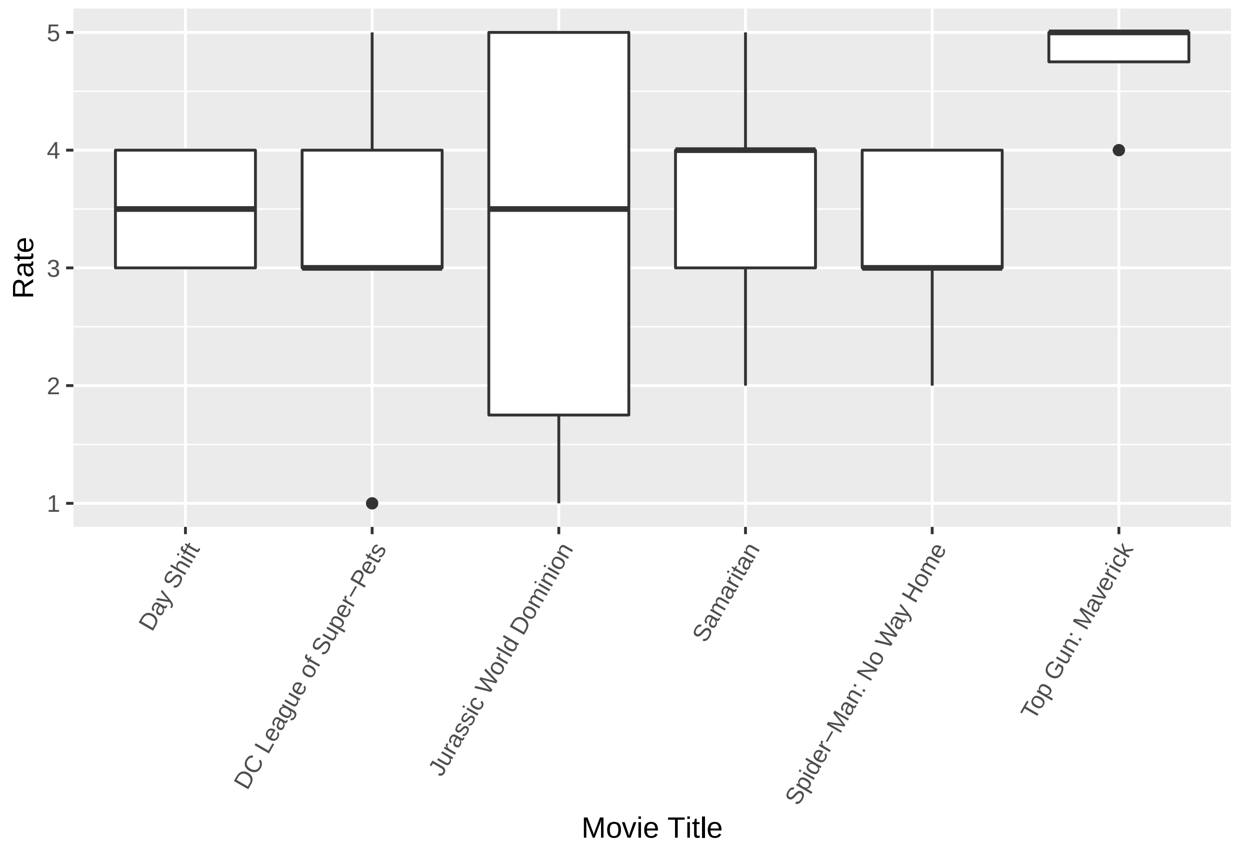
```
dfmedian <- dftmp %>%
        group_by(movie_id) %>%
        mutate(median = median(rate))
```

```r
dfmedian$rate <- ifelse(dfmedian$rate == 0, dfmedian$median, dfmedian$rate)

print(dfmedian, n=100)
```

```
## # A tibble: 27 x 6
## # Groups:   movie_id [6]
##    movie_id email_address           rate popularity title                  median
##       <int> <chr>                  <int>      <dbl> <chr>                   <dbl>
##  1   361743 blacksmilez@gmail.com      4      2030. Top Gun: Maverick         5
##  2   361743 negativetae@gmail.com      5      2030. Top Gun: Maverick         5
##  3   361743 nury95@hotmail.com         5      2030. Top Gun: Maverick         5
##  4   361743 s88724@gmail.com           5      2030. Top Gun: Maverick         5
##  5   507086 blacksmilez@gmail.com      5      2084. Jurassic World Domin~    3.5
##  6   507086 negativetae@gmail.com      1      2084. Jurassic World Domin~    3.5
##  7   507086 nury95@hotmail.com         5      2084. Jurassic World Domin~    3.5
##  8   507086 s88724@gmail.com           2      2084. Jurassic World Domin~    3.5
##  9   539681 blacksmilez@gmail.com      3      2738. DC League of Super-P~    3
## 10   539681 negativetae@gmail.com      1      2738. DC League of Super-P~    3
## 11   539681 nury95@hotmail.com         5      2738. DC League of Super-P~    3
## 12   539681 delight_32@hotmail.com     4      2738. DC League of Super-P~    3
## 13   539681 s88724@gmail.com           3      2738. DC League of Super-P~    3
## 14   629176 blacksmilez@gmail.com      2      5115. Samaritan                4
## 15   629176 negativetae@gmail.com      3      5115. Samaritan                4
## 16   629176 nury95@hotmail.com         5      5115. Samaritan                4
## 17   629176 delight_32@hotmail.com     4      5115. Samaritan                4
## 18   629176 s88724@gmail.com           4      5115. Samaritan                4
## 19   634649 blacksmilez@gmail.com      4      1142. Spider-Man: No Way H~    3
## 20   634649 negativetae@gmail.com      2      1142. Spider-Man: No Way H~    3
## 21   634649 nury95@hotmail.com         3      1142. Spider-Man: No Way H~    3
## 22   634649 delight_32@hotmail.com     3      1142. Spider-Man: No Way H~    3
## 23   634649 s88724@gmail.com           4      1142. Spider-Man: No Way H~    3
## 24   755566 negativetae@gmail.com      4      1403. Day Shift              3.5
## 25   755566 nury95@hotmail.com         3      1403. Day Shift              3.5
## 26   755566 delight_32@hotmail.com     4      1403. Day Shift              3.5
## 27   755566 s88724@gmail.com           3      1403. Day Shift              3.5
```

**Graph of missing values filled with *median* values**

```
ggplot(dfmedian, aes(x=title, y=rate)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1.0 )) +
  labs(x='Movie Title', y='Rate')
```



## Missing Data (3)

**For calibration, the missing values can be filled with *max* values.**

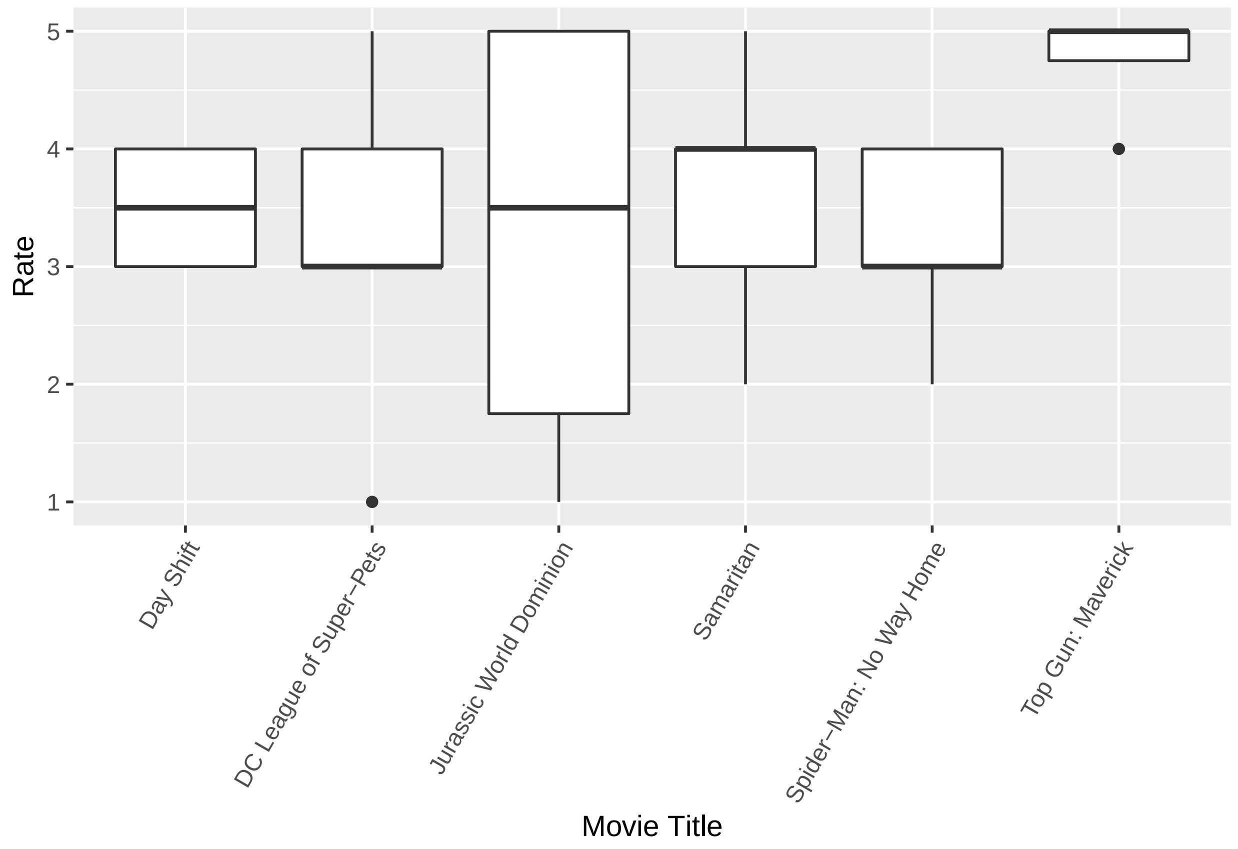
```
dfmax <- dftmp %>%
        group_by(movie_id) %>%
        mutate(max = max(rate))


dfmax$rate <- ifelse(dfmax$rate == 0, dfmax$max, dfmax$rate)
```

```r
print(dfmax, n=100)
```

```
## # A tibble: 27 x 6
## # Groups:   movie_id [6]
##    movie_id email_address           rate popularity title                    max
##       <int> <chr>                  <int>      <dbl> <chr>                  <int>
##  1   361743 blacksmilez@gmail.com      4      2030. Top Gun: Maverick          5
##  2   361743 negativetae@gmail.com      5      2030. Top Gun: Maverick          5
##  3   361743 nury95@hotmail.com         5      2030. Top Gun: Maverick          5
##  4   361743 s88724@gmail.com           5      2030. Top Gun: Maverick          5
##  5   507086 blacksmilez@gmail.com      5      2084. Jurassic World Domini~     5
##  6   507086 negativetae@gmail.com      1      2084. Jurassic World Domini~     5
##  7   507086 nury95@hotmail.com         5      2084. Jurassic World Domini~     5
##  8   507086 s88724@gmail.com           2      2084. Jurassic World Domini~     5
##  9   539681 blacksmilez@gmail.com      3      2738. DC League of Super-Pe~     5
## 10   539681 negativetae@gmail.com      1      2738. DC League of Super-Pe~     5
## 11   539681 nury95@hotmail.com         5      2738. DC League of Super-Pe~     5
## 12   539681 delight_32@hotmail.com     4      2738. DC League of Super-Pe~     5
## 13   539681 s88724@gmail.com           3      2738. DC League of Super-Pe~     5
## 14   629176 blacksmilez@gmail.com      2      5115. Samaritan                  5
## 15   629176 negativetae@gmail.com      3      5115. Samaritan                  5
## 16   629176 nury95@hotmail.com         5      5115. Samaritan                  5
## 17   629176 delight_32@hotmail.com     4      5115. Samaritan                  5
## 18   629176 s88724@gmail.com           4      5115. Samaritan                  5
## 19   634649 blacksmilez@gmail.com      4      1142. Spider-Man: No Way Ho~     4
## 20   634649 negativetae@gmail.com      2      1142. Spider-Man: No Way Ho~     4
## 21   634649 nury95@hotmail.com         3      1142. Spider-Man: No Way Ho~     4
## 22   634649 delight_32@hotmail.com     3      1142. Spider-Man: No Way Ho~     4
## 23   634649 s88724@gmail.com           4      1142. Spider-Man: No Way Ho~     4
## 24   755566 negativetae@gmail.com      4      1403. Day Shift                  4
## 25   755566 nury95@hotmail.com         3      1403. Day Shift                  4
## 26   755566 delight_32@hotmail.com     4      1403. Day Shift                  4
## 27   755566 s88724@gmail.com           3      1403. Day Shift                  4
```

**Graph of missing values filled with *max* values**

```
ggplot(dfmax, aes(x=title, y=rate)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1.0 )) +
  labs(x='Movie Title', y='Rate')
```



## Global Baseline Estimate

```
df <- fetch(dbSendQuery(con, 'select movie_id, email_address, rate from survey_result where survey_id =
dbClearResult(dbListResults(con)[[1]])
```

```
## [1] TRUE
```

There is no record for the review with no rating in the table because Null values are not stored in the data for data integrity. Therefore, use pivot_wider() and pivot_longer() function to create missing record. ### Step 1. Use the pivot_wider() function to create a data frame identical to Excel format. Missing record will display as a 'NA'.

```
pv_wider <- df %>%
              pivot_wider(
                names_from = movie_id,
                values_from = rate
              )

print(pv_wider)
```

```
## # A tibble: 5 x 7
##   email_address          `361743` `507086` `634649` `539681` `629176` `755566`
##   <chr>                     <int>    <int>    <int>    <int>    <int>    <int>
## 1 blacksmilez@gmail.com         4        5        4        3        2       NA
## 2 negativetae@gmail.com         5        1        2        1        3        4
## 3 nury95@hotmail.com            5        5        3        5        5        3
## 4 s88724@gmail.com              5        2        4        3        4        3
## 5 delight_32@hotmail.com       NA       NA        3        4        4        4
```

**Step 2. If change the data frame created in step 1 to its original form using the pivot_longer() function, a data frame including the missing record is created.**

```
pv_longer <- pv_wider %>%
              pivot_longer(
                cols = colnames(pv_wider)[2:7],
                names_to = 'movie_id',
                values_to = 'rate'
              )

print(pv_longer)
```

```
## # A tibble: 30 x 3
##    email_address         movie_id  rate
##    <chr>                 <chr>    <int>
##  1 blacksmilez@gmail.com 361743       4
##  2 blacksmilez@gmail.com 507086       5
##  3 blacksmilez@gmail.com 634649       4
##  4 blacksmilez@gmail.com 539681       3
##  5 blacksmilez@gmail.com 629176       2
##  6 blacksmilez@gmail.com 755566      NA
##  7 negativetae@gmail.com 361743       5
##  8 negativetae@gmail.com 507086       1
##  9 negativetae@gmail.com 634649       2
## 10 negativetae@gmail.com 539681       1
## # ... with 20 more rows
```

**Step 3. use is.na() function to exclude NA value. After that calculate movie_avg for each movie. (identical to Excel A18:G18)**

```
movie_avg <- pv_longer %>%
                filter(!is.na(rate)) %>%
                group_by(movie_id) %>%
                summarise(movie_avg = mean(rate))
print(movie_avg)
```

```
## # A tibble: 6 x 2
##   movie_id movie_avg
##   <chr>        <dbl>
## 1 361743        4.75
## 2 507086        3.25
## 3 539681        3.2
## 4 629176        3.6
## 5 634649        3.2
## 6 755566        3.5
```

**Step 4. use is.na() function to exclude NA value. After that calculate movie_mean. (identical to Excel H18)**

```
movie_mean <- mean(pv_longer[!is.na(pv_longer$rate), ]$rate)


print(movie_mean)
```

```
## [1] 3.555556
```

**Step 5. Using mutate, add column named sub_avg_mean in the movie_avg created in step 3 and insert movie_mean subtacted from movie_avg. (identical to Excel B19:G19)**

```
movie_compute <- movie_avg %>%
                    mutate(subs_avg_mean = movie_avg - movie_mean)
```

```r
print(movie_compute)
```

```
## # A tibble: 6 x 3
##   movie_id movie_avg subs_avg_mean
##   <chr>        <dbl>         <dbl>
## 1 361743        4.75        1.19
## 2 507086        3.25       -0.306
## 3 539681        3.2        -0.356
## 4 629176        3.6         0.0444
## 5 634649        3.2        -0.356
## 6 755566        3.5        -0.0556
```

**Step 6. use is.na() function to exclude NA value. After that calculate each person's user__avg and user__avg__mean__movie (identical to Excel H2:17, I2:17)**

```r
user_compute <- pv_longer %>%
                  filter(!is.na(rate)) %>%
                  group_by(email_address) %>%
                  mutate(
                    user_avg = mean(rate),
                    sub_user_avg_mean_movie = mean(rate) - movie_mean
                  )

print(user_compute)
```

```
## # A tibble: 27 x 5
## # Groups:   email_address [5]
##    email_address         movie_id  rate user_avg sub_user_avg_mean_movie
##    <chr>                 <chr>    <int>    <dbl>                   <dbl>
##  1 blacksmilez@gmail.com 361743       4     3.6                   0.0444
##  2 blacksmilez@gmail.com 507086       5     3.6                   0.0444
##  3 blacksmilez@gmail.com 634649       4     3.6                   0.0444
##  4 blacksmilez@gmail.com 539681       3     3.6                   0.0444
##  5 blacksmilez@gmail.com 629176       2     3.6                   0.0444
##  6 negativetae@gmail.com 361743       5     2.67                 -0.889
##  7 negativetae@gmail.com 507086       1     2.67                 -0.889
##  8 negativetae@gmail.com 634649       2     2.67                 -0.889
##  9 negativetae@gmail.com 539681       1     2.67                  -0.889
## 10 negativetae@gmail.com 629176       3     2.67                  -0.889
## # ... with 17 more rows
```

**Step 7. use is.na()function to bring record with NA value and merge(m1) with the movie_compute crated in step 5 (same as Join in SQL). Then, select 'email_address', 'user_avg', 'sub_user_avg_mean_movie' from user_compute in step 6 to distinct using unique() function for merge with m1. Calculate rate to insert.**

```
m1 <- merge(pv_longer[is.na(pv_longer$rate),], movie_compute)
print(m1)
```

```
##   movie_id          email_address rate movie_avg subs_avg_mean
## 1   361743 delight_32@hotmail.com   NA      4.75    1.19444444
## 2   507086 delight_32@hotmail.com   NA      3.25   -0.30555556
## 3   755566  blacksmilez@gmail.com   NA      3.50   -0.05555556
```

```
m2 <- merge(m1,
            unique(user_compute[,c('email_address', 'user_avg', 'sub_user_avg_mean_movie')]),
            by.x=c('email_address'),
            by.y=c('email_address')) %>%
      mutate(rate = round(movie_mean + subs_avg_mean + sub_user_avg_mean_movie, 0)) %>%
      select('email_address', 'movie_id', 'rate', 'user_avg', 'sub_user_avg_mean_movie')
print(m2)
```

```
##            email_address movie_id rate user_avg sub_user_avg_mean_movie
## 1  blacksmilez@gmail.com   755566    4     3.60              0.04444444
## 2 delight_32@hotmail.com   361743    5     3.75              0.19444444
## 3 delight_32@hotmail.com   507086    3     3.75              0.19444444
```

**Step 8. Combine m2 created in step 7 and user_compute with no NA record using the union() function. Then merges with dfMovies so that the title can be displayed instead of the movie_id.**

```
final <- merge(union(user_compute, m2),
               dfMovies) %>%
          select('email_address', 'title', 'rate', 'user_avg',  'sub_user_avg_mean_movie')
print(final)
```

```
##              email_address                    title rate user_avg
## 1   blacksmilez@gmail.com       Top Gun: Maverick    4 3.600000
## 2   negativetae@gmail.com       Top Gun: Maverick    5 2.666667
## 3      nury95@hotmail.com       Top Gun: Maverick    5 4.333333
## 4  delight_32@hotmail.com       Top Gun: Maverick    5 3.750000
## 5        s88724@gmail.com       Top Gun: Maverick    5 3.500000
## 6   negativetae@gmail.com Jurassic World Dominion    1 2.666667
## 7        s88724@gmail.com Jurassic World Dominion    2 3.500000
## 8      nury95@hotmail.com Jurassic World Dominion    5 4.333333
## 9   blacksmilez@gmail.com Jurassic World Dominion    5 3.600000
## 10 delight_32@hotmail.com Jurassic World Dominion    3 3.750000
## 11  negativetae@gmail.com DC League of Super-Pets    1 2.666667
## 12     nury95@hotmail.com DC League of Super-Pets    5 4.333333
## 13       s88724@gmail.com DC League of Super-Pets    3 3.500000
## 14  blacksmilez@gmail.com DC League of Super-Pets    3 3.600000
## 15 delight_32@hotmail.com DC League of Super-Pets    4 3.750000
## 16  negativetae@gmail.com               Samaritan    3 2.666667
## 17     nury95@hotmail.com               Samaritan    5 4.333333
## 18  blacksmilez@gmail.com               Samaritan    2 3.600000
## 19       s88724@gmail.com               Samaritan    4 3.500000
## 20 delight_32@hotmail.com               Samaritan    4 3.750000
## 21  negativetae@gmail.com Spider-Man: No Way Home    2 2.666667
## 22  blacksmilez@gmail.com Spider-Man: No Way Home    4 3.600000
## 23       s88724@gmail.com Spider-Man: No Way Home    4 3.500000
## 24 delight_32@hotmail.com Spider-Man: No Way Home    3 3.750000
## 25     nury95@hotmail.com Spider-Man: No Way Home    3 4.333333
## 26  negativetae@gmail.com               Day Shift    4 2.666667
## 27       s88724@gmail.com               Day Shift    3 3.500000
## 28 delight_32@hotmail.com               Day Shift    4 3.750000
## 29  blacksmilez@gmail.com               Day Shift    4 3.600000
## 30     nury95@hotmail.com               Day Shift    3 4.333333
##    sub_user_avg_mean_movie
## 1               0.04444444
## 2              -0.88888889
## 3               0.77777778
## 4               0.19444444
## 5               -0.05555556
## 6               -0.88888889
## 7               -0.05555556
## 8                0.77777778
## 9                0.04444444
## 10               0.19444444
## 11              -0.88888889
## 12               0.77777778
## 13              -0.05555556
## 14               0.04444444
## 15               0.19444444
## 16              -0.88888889
## 17               0.77777778
## 18               0.04444444
## 19              -0.05555556
## 20               0.19444444
## 21              -0.88888889
## 22               0.04444444
## 23              -0.05555556
## 24               0.19444444
## 25               0.77777778
## 26              -0.88888889
## 27              -0.05555556
## 28               0.19444444
## 29               0.04444444
## 30               0.77777778
```

**step 9: use pivot_wider() function to make the data frame created in step 8 identical to Excel format and display on the screen. Use the Excel formula to review the value.**

```
final %>%
  pivot_wider(
    names_from = title,
    values_from = rate
  ) %>%
  select(1, 4:9, 2, 3)
```

```
## # A tibble: 5 x 9
##   email_address  Top G~1 Juras~2 DC Le~3 Samar~4 Spide~5 Day S~6 user_~7 sub_u~8
##   <chr>            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 blacksmilez@g~       4       5       3       2       4       4    3.6   0.0444
## 2 negativetae@g~       5       1       1       3       2       4    2.67 -0.889
## 3 nury95@hotmai~       5       5       5       5       3       3    4.33  0.778
## 4 delight_32@ho~       5       3       4       4       3       4    3.75  0.194
## 5 s88724@gmail.~       5       2       3       4       4       3    3.5  -0.0556
## # ... with abbreviated variable names 1: `Top Gun: Maverick`,
## #   2: `Jurassic World Dominion`, 3: `DC League of Super-Pets`, 4: Samaritan,
## #   5: `Spider-Man: No Way Home`, 6: `Day Shift`, 7: user_avg,
## #   8: sub_user_avg_mean_movie
```

```
# mean_movie
movie_mean
```

```
## [1] 3.555556
```

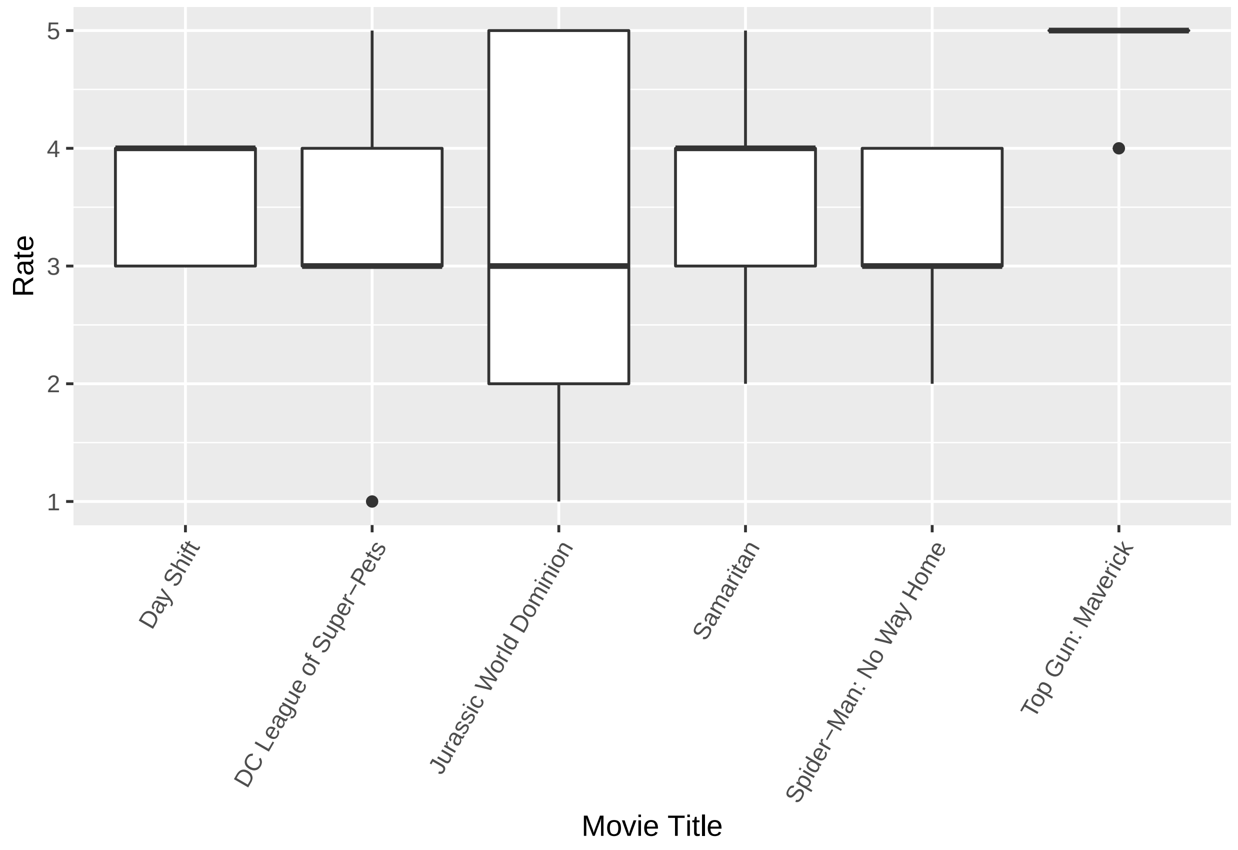
```
# movie_avg
merge(movie_compute, dfMovies) %>%
  select('title', 'movie_avg') %>%
  pivot_wider(
    names_from = title,
    values_from = movie_avg
  )
```

```
## # A tibble: 1 x 6
##   `Top Gun: Maverick` `Jurassic World Dominion` DC Lea~1 Samar~2 Spide~3 Day S~4
##                 <dbl>                     <dbl>    <dbl>   <dbl>   <dbl>   <dbl>
## 1                4.75                      3.25      3.2     3.6     3.2     3.5
## # ... with abbreviated variable names 1: `DC League of Super-Pets`,
## #   2: Samaritan, 3: `Spider-Man: No Way Home`, 4: `Day Shift`
```

```
# avg-mean
merge(movie_compute, dfMovies) %>%
  select('title', 'subs_avg_mean') %>%
  pivot_wider(
    names_from = title,
    values_from = subs_avg_mean
)
```

```
## # A tibble: 1 x 6
##   `Top Gun: Maverick` `Jurassic World Dominion` DC Lea~1 Samar~2 Spide~3 Day S~4
##                 <dbl>                     <dbl>    <dbl>   <dbl>   <dbl>   <dbl>
## 1                1.19                    -0.306   -0.356  0.0444  -0.356 -0.0556
## # ... with abbreviated variable names 1: `DC League of Super-Pets`,
## #   2: Samaritan, 3: `Spider-Man: No Way Home`, 4: `Day Shift`
```

```
ggplot(final, aes(x=title, y=rate)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1.0 )) +
  labs(x='Movie Title', y='Rate')
```



## Closure

The advantage of normalization is that it does not have unnecessary redundant data. It is possible to maintain the integrity of the data by removing the duplicate data. This is a big advantage of relational databases, but in other words, it can also be a big disadvantage. This is because emphasizing excessive normalization causes problems in system performance. Data standardization increases mutual communication by further specifying data. There are various ways to process missing data, but I checked by filling it with mean, median, and max values. Each has a slight difference, so I think we should choose and use it as needed.

Github: https://github.com/blacksmilez/DATA607/tree/main/Assignment02