Week2 Assignment: SQL and R

Fomba Kassoh

2023-15-08

Assignment – SQL and R

Overview

The codes below loads movie ratings data from the csv file movies_data. The data was collected from classmates and friends using a google form. The following six movies were presented in the survey: (1) Barbie, (2) Black Panther Wakanda, (3) Oppenheimer, (4) Spiderman, (5) Top Gun Maverick, (6) The Nun II. Each respondent was asked to rate the movies they have watched based on the following factors: Entertainment value, story, animation/visuals, emotional beats, and humor.

The data collected was downloaded into a csv file (movies_data.csv). A database, movies, was created in MySQL and the survey data loaded from movies_data.csv into the database table, movies_data. In addition, a movies database table was created and loaded with information from the movies_info.csv file. A critics database table was also created and loaded with the information of the survey respondents.

The respondent to the survey, called critics were asked to rate

Link to article:https://projects.fivethirtyeight.com/coronavirus-polls/ link to data frame: https://raw.githubusercontent.com/hawa1983/Week1_Assignment/main/covid_approval_polls_adjusted.csv

Load the relevant libraries

We start by installing the relevant packages and loading the libraries as below

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(dplyr)
library(RMySQL)
```

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

```r
library(DBI)
library(readr)
library(keyring)
```

Create the database connection object

Here, we connect to the MySQL database.

```r
db_host <- "localhost"
db_port <- 5432
db_user <- "root"
db_password <- "7!8Kas33!4"
movies <- "movies"


# Create a connection object
con <- dbConnect(MySQL(),
                 user = "root",
                 password = "7!8Kas33!4",
                 host = "localhost",
                 name = "movies")

cat("Connected to database successfully.")
```

```
## Connected to database successfully.
```

Preview the data

Next we create the movies database. Dropping the database is done so that the code will reproduce the steps taken.

```r
# if exist drop movies database
dbSendQuery(con, "DROP DATABASE IF EXISTS movies")
```

```
## <MySQLResult:0,0,0>
```

```r
# Create the MySQL database
dbExecute(con, "CREATE DATABASE IF NOT EXISTS movies")
```

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

```r
# Select the movies database as the default database
dbSendQuery(con, "USE movies")
```

```
## <MySQLResult:289741304,0,2>
```

```r
# List the databases to verify
dbListTables(con)
```

```
## character(0)
```

```
cat("Database created successfully.")
```

```
## Database created successfully.
```

Drop the database tables if they exist

Here we drop the database tables so the code can reproduce them.

```
tables_to_drop <- c("movies_data", "ratings", "critics", "factors", "movies", "joint_table")

for (table_name in tables_to_drop) {
  if (dbExistsTable(con, table_name)) {
    query <- paste("DROP TABLE", table_name)
    dbExecute(con, query)
  }
}
```

Create the database tables

Now we create the database tables

```
create_table_query <- "
CREATE TABLE IF NOT EXISTS movies_data (
  time_stamp VARCHAR(255) NOT NULL,
  critic VARCHAR(255) NOT NULL,
  age_range VARCHAR(255) NOT NULL,
  movie VARCHAR(255) NOT NULL,
  entertainment_value VARCHAR(255) NOT NULL,
  story VARCHAR(255) NOT NULL,
  animation_visuals VARCHAR(255) NOT NULL,
  emotional_beats VARCHAR(255) NOT NULL,
  humor VARCHAR(255) NOT NULL
)
"
dbExecute(con, create_table_query)
```

```
## [1] 0
```

```
create_table_query <- "
CREATE TABLE IF NOT EXISTS movies (
  movie_id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  release_date VARCHAR(255) NOT NULL,
  runnunig_time INT NOT NULL,
  budget DOUBLE NOT NULL,
  box_office DOUBLE NOT NULL
)
"
dbExecute(con, create_table_query)
```

```
## [1] 0
```

```
create_table_query <- "
CREATE TABLE IF NOT EXISTS factors (
  factor_id INT AUTO_INCREMENT PRIMARY KEY,
  entertainment_value VARCHAR(255) NOT NULL,
  story VARCHAR(255) NOT NULL,
  animation_visuals VARCHAR(255) NOT NULL,
  emotional_beats VARCHAR(255) NOT NULL,
  humor VARCHAR(255) NOT NULL
)
"
dbExecute(con, create_table_query)
```

```
## [1] 0
```

```
create_table_query <- "
CREATE TABLE IF NOT EXISTS critics (
  name VARCHAR(255) NOT NULL,
  age_range VARCHAR(255) NOT NULL,
  critic_id INT PRIMARY KEY
)
"
dbExecute(con, create_table_query)
```

```
## [1] 0
```

```
create_table_query <- "
CREATE TABLE IF NOT EXISTS joint_table (
  critic_id INT NOT NULL,
  movie_id INT NOT NULL,
  name VARCHAR(255) NOT NULL,
  movie VARCHAR(255) NOT NULL,
  FOREIGN KEY (critic_id) REFERENCES critics(critic_id),
  FOREIGN KEY (movie_id) REFERENCES movies(movie_id)
)
"
dbExecute(con, create_table_query)
```

```
## [1] 0
```

```
cat("Database tables successfully.")
```

```
## Database tables successfully.
```

read the movie survey csv file into a data frame.

The movie ratings survey was saved in the following directory: C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/

```r
movies_data <- read_csv('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movies_data.csv')
```

```
## Rows: 60 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): timestamp, name, age_range, movie
## dbl (5): entertainment_value, story, animation_visuals, emotional_beats, humor
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```r
# preview data
glimpse(movies_data)
```

```
## Rows: 60
## Columns: 9
## $ timestamp           <chr> "9/14/2023 16:13", "9/14/2023 16:14", "9/14/2023 1~
## $ name                <chr> "Tony", "Jean", "Kristin", "Shamecca", "Shamecca",~
## $ age_range           <chr> "50+", "20 - 29", "20 - 29", "20 - 29", "20 - 29",~
## $ movie               <chr> "Top Gun Maverick", "Barbie", "Barbie", "Black Pan~
## $ entertainment_value <dbl> 5, 5, 5, 5, 4, 5, 4, 5, 5, 3, 5, 5, 5, 5, 5, 4, 5,~
## $ story               <dbl> 4, 3, NA, 5, 3, 3, 2, 5, 5, 3, 5, 5, 5, 5, 5, 2, 5~
## $ animation_visuals   <dbl> 5, 4, 5, 5, 5, 5, 3, 3, 5, 3, 5, 5, 5, 5, 5, 5, 5,~
## $ emotional_beats     <dbl> 4, 3, 5, 5, 1, 3, 3, 4, 5, 3, 5, 5, 4, 5, NA, 5, 5~
## $ humor               <dbl> 4, 4, 5, 2, 3, 5, 3, 2, 5, 4, 5, 5, 1, 5, 3, 4, 4,~
```

Ignore missing values in calculation

The preview of the movies_data above shows that there are missing values. If we find the average rating of a critis across all the factors rated, we must ignore the missing value in order for the average to be correct.

```r
average_rating <- movies_data |> mutate(Avg_rating = mean(c(entertainment_value, story, animation_visual

average_rating
```

```
## # A tibble: 60 x 10
##    timestamp   name  age_range movie entertainment_value story animation_visuals
##    <chr>       <chr> <chr>     <chr>               <dbl> <dbl>             <dbl>
##  1 9/14/2023 ~ Tony  50+       Top ~                   5     4                 5
##  2 9/14/2023 ~ Jean  20 - 29   Barb~                   5     3                 4
##  3 9/14/2023 ~ Kris~ 20 - 29   Barb~                   5    NA                 5
##  4 9/14/2023 ~ Sham~ 20 - 29   Blac~                   5     5                 5
##  5 9/14/2023 ~ Sham~ 20 - 29   Barb~                   4     3                 5
##  6 9/14/2023 ~ Sham~ 20 - 29   Spid~                   5     3                 5
##  7 9/14/2023 ~ Kossi 20 - 29   Barb~                   4     2                 3
##  8 9/14/2023 ~ Sean~ 30 - 39   Oppe~                   5     5                 3
##  9 9/14/2023 ~ Dom   15 - 19   Spid~                   5     5                 5
## 10 9/14/2023 ~ Sia   20 - 29   Barb~                   3     3                 3
## # i 50 more rows
## # i 3 more variables: emotional_beats <dbl>, humor <dbl>, Avg_rating <dbl>
```

Replace missing values with 0

```
movies_data_1 <- movies_data |>
  mutate(entertainment_value = replace_na(entertainment_value, 0),
         story = replace_na(story, 0),
         animation_visuals = replace_na(animation_visuals, 0),
         emotional_beats = replace_na(emotional_beats, 0),
         humor = replace_na(humor, 0)
         )

movies_data_1
```

```
## # A tibble: 60 x 9
##    timestamp   name  age_range movie entertainment_value story animation_visuals
##    <chr>       <chr> <chr>     <chr>               <dbl> <dbl>             <dbl>
##  1 9/14/2023 ~ Tony  50+       Top ~                   5     4                 5
##  2 9/14/2023 ~ Jean  20 - 29   Barb~                   5     3                 4
##  3 9/14/2023 ~ Kris~ 20 - 29   Barb~                   5     0                 5
##  4 9/14/2023 ~ Sham~ 20 - 29   Blac~                   5     5                 5
##  5 9/14/2023 ~ Sham~ 20 - 29   Barb~                   4     3                 5
##  6 9/14/2023 ~ Sham~ 20 - 29   Spid~                   5     3                 5
##  7 9/14/2023 ~ Kossi 20 - 29   Barb~                   4     2                 3
##  8 9/14/2023 ~ Sean~ 30 - 39   Oppe~                   5     5                 3
##  9 9/14/2023 ~ Dom   15 - 19   Spid~                   5     5                 5
## 10 9/14/2023 ~ Sia   20 - 29   Barb~                   3     3                 3
## # i 50 more rows
## # i 2 more variables: emotional_beats <dbl>, humor <dbl>
```

Replace NA in column A with the mean of column A

We can also replace the missing values with the mean of the rating

```
movies_data <- movies_data %>%
  mutate(
    entertainment_value = ifelse(is.na(entertainment_value), mean(entertainment_value, na.rm = TRUE), en
    story = ifelse(is.na(story), mean(story, na.rm = TRUE), story),
    animation_visuals = ifelse(is.na(animation_visuals), mean(animation_visuals, na.rm = TRUE), animati
    emotional_beats = ifelse(is.na(emotional_beats), mean(emotional_beats, na.rm = TRUE), emotional_bea
    humor = ifelse(is.na(humor), mean(humor, na.rm = TRUE), humor),
    )

movies_data
```

```
## # A tibble: 60 x 9
##    timestamp   name  age_range movie entertainment_value story animation_visuals
##    <chr>       <chr> <chr>     <chr>               <dbl> <dbl>             <dbl>
##  1 9/14/2023 ~ Tony  50+       Top ~                   5  4                    5
##  2 9/14/2023 ~ Jean  20 - 29   Barb~                   5  3                    4
##  3 9/14/2023 ~ Kris~ 20 - 29   Barb~                   5  4.17                 5
##  4 9/14/2023 ~ Sham~ 20 - 29   Blac~                   5  5                    5
##  5 9/14/2023 ~ Sham~ 20 - 29   Barb~                   4  3                    5
##  6 9/14/2023 ~ Sham~ 20 - 29   Spid~                   5  3                     5
##  7 9/14/2023 ~ Kossi 20 - 29   Barb~                   4  2                     3
##  8 9/14/2023 ~ Sean~ 30 - 39   Oppe~                   5  5                     3
##  9 9/14/2023 ~ Dom   15 - 19   Spid~                   5  5                     5
## 10 9/14/2023 ~ Sia   20 - 29   Barb~                   3  3                     3
## # i 50 more rows
## # i 2 more variables: emotional_beats <dbl>, humor <dbl>
```

Change the data types to the appopiate data type

Change the timestamp data type to datetime. Change the movie data type to factors, and the ratings to integer

```
movies_data <- read_csv('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movies_data.csv',
                        col_types = cols(
                   timestamp = col_datetime(format = "%m/%d/%Y %H:%M")
                 ))

movies_data <- movies_data |>
  mutate(
    timestamp = as_datetime(timestamp),
    movie = as_factor(movie),
    entertainment_value = as.integer(entertainment_value),
    story = as.integer(story),
    animation_visuals = as.integer(animation_visuals),
    emotional_beats = as.integer(emotional_beats),
    humor = as.integer(humor)
    )

# overwrite the data back to the movies_data CSV file to persist the changes
#write.csv(movies_data, "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movies_data.csv", row.names = FA

glimpse(movies_data)
```

```
## Rows: 60
## Columns: 9
## $ timestamp           <dttm> 2023-09-14 16:13:00, 2023-09-14 16:14:00, 2023-09~
## $ name                <chr> "Tony", "Jean", "Kristin", "Shamecca", "Shamecca",~
## $ age_range           <chr> "50+", "20 - 29", "20 - 29", "20 - 29", "20 - 29",~
## $ movie               <fct> Top Gun Maverick, Barbie, Barbie, Black Panther Wa~
## $ entertainment_value <int> 5, 5, 5, 5, 4, 5, 4, 5, 5, 3, 5, 5, 5, 5, 5, 4, 5,~
## $ story               <int> 4, 3, NA, 5, 3, 3, 2, 5, 5, 3, 5, 5, 5, 5, 5, 2, 5~
## $ animation_visuals   <int> 5, 4, 5, 5, 5, 5, 3, 3, 5, 3, 5, 5, 5, 5, 5, 5, 5,~
## $ emotional_beats     <int> 4, 3, 5, 5, 1, 3, 3, 4, 5, 3, 5, 5, 4, 5, NA, 5, 5~
## $ humor               <int> 4, 4, 5, 2, 3, 5, 3, 2, 5, 4, 5, 5, 1, 5, 3, 4, 4,~
```

Basic information about each movie

Some basic information about each movie was collected from Wikipedia and saved in a csv file. The file is read into movies data frame below. The appropiate data type is assigned and the csv file overwritten to persist the changes.

```
movies_df <- read_csv('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movies.csv',
                        col_types = cols(
                   release_date = col_date(format = "%m/%d/%Y")
                 ))

movies_df <- movies_df |>
  mutate(
    id = as.integer(id),
    movie = as_factor(name),
    running_time = as.integer(running_time),
    budget = as.double(budget),
    box_office = as.double(box_office)
    )

movies_df
```

```
## # A tibble: 6 x 7
##      id name                  release_date running_time budget box_office movie
##   <int> <chr>                 <date>              <int>  <dbl>      <dbl> <fct>
## 1     1 Barbie                2023-07-09            114  145         1.41 Barbie
## 2     2 Black Panther Wakanda 2022-10-26            161  260         0.86 Black~
## 3     3 Oppenheimer           2023-07-11            180  100         0.9  Oppen~
## 4     4 Spiderman             2021-12-13            148  200         1.92 Spide~
## 5     5 Top Gun Maverick      2022-04-28            130  177         1.5  Top G~
## 6     6 The Nun II            2023-09-08            110   38.5       0.1  The N~
```

Movie Critics

A MySQL table is created for the movie critics. To get the data for the table, we create a csv file by getting the unique values from the movies_data data frame. An id variable is added to get a unique ID for each critic

```
# Select unique values from a column
critics <- movies_data |>
  distinct(name, age_range)

critics <- critics |>
  mutate(id = c(1:nrow(critics)))


critics
```

```
## # A tibble: 44 x 3
##    name       age_range    id
##    <chr>      <chr>     <int>
##  1 Tony       50+           1
##  2 Jean       20 - 29       2
##  3 Kristin    20 - 29       3
##  4 Shamecca   20 - 29       4
##  5 Kossi      20 - 29       5
##  6 Sean Amato 30 - 39       6
##  7 Dom        15 - 19       7
##  8 Sia        20 - 29       8
##  9 Dustin     15 - 19       9
## 10 Zainab     20 - 29      10
## # i 34 more rows
```

Create a joint table with critic ID and movie ID

To do this, we create a left joint and select the critic and movie ids. The we load the data frame into the joint

```
# Perform the operation
joined_df <- movies_data |>
  left_join(critics, by = "name") |>
  left_join(movies_df, by = "movie")

joined_df
```

```
## # A tibble: 60 x 17
##    timestamp           name.x     age_range.x movie    entertainment_value story
##    <dttm>              <chr>      <chr>       <fct>                  <int> <int>
##  1 2023-09-14 16:13:00 Tony       50+         Top Gun~                   5     4
##  2 2023-09-14 16:14:00 Jean       20 - 29     Barbie                     5     3
##  3 2023-09-14 16:24:00 Kristin    20 - 29     Barbie                     5    NA
##  4 2023-09-14 16:24:00 Shamecca   20 - 29     Black P~                   5     5
##  5 2023-09-14 16:24:00 Shamecca   20 - 29     Barbie                     4     3
##  6 2023-09-14 16:25:00 Shamecca   20 - 29     Spiderm~                   5     3
##  7 2023-09-14 16:27:00 Kossi      20 - 29     Barbie                     4     2
##  8 2023-09-14 16:29:00 Sean Amato 30 - 39     Oppenhe~                   5     5
##  9 2023-09-14 17:40:00 Dom        15 - 19     Spiderm~                   5     5
## 10 2023-09-14 17:48:00 Sia        20 - 29     Barbie                     3     3
## # i 50 more rows
## # i 11 more variables: animation_visuals <int>, emotional_beats <int>,
## #   humor <int>, age_range.y <chr>, id.x <int>, id.y <int>, name.y <chr>,
## #   release_date <date>, running_time <int>, budget <dbl>, box_office <dbl>
```

```
joint_id_df <- joined_df |>
  select(critic_id = id.x, movie_id = id.y, critic = name.x, movie = name.y)

write_csv(joint_id_df, "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/joint_table.csv")
joint_id_df <- read_csv("C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/joint_table.csv")
```

```
## Rows: 60 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): critic, movie
## dbl (2): critic_id, movie_id
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
joint_id_df
```

```
## # A tibble: 60 x 4
##    critic_id movie_id critic     movie
##        <dbl>    <dbl> <chr>      <chr>
##  1         1        5 Tony       Top Gun Maverick
##  2         2        1 Jean       Barbie
##  3         3        1 Kristin    Barbie
##  4         4        2 Shamecca   Black Panther Wakanda
##  5         4        1 Shamecca   Barbie
##  6         4        4 Shamecca   Spiderman
##  7         5        1 Kossi      Barbie
##  8         6        3 Sean Amato Oppenheimer
##  9         7        4 Dom        Spiderman
## 10         8        1 Sia        Barbie
## # i 50 more rows
```

Load the data frames into the MySQL tables

Now we will load the csv files into the MySQL tables.

```
load_movies_data_query <- "
  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movies_data.csv'
  INTO TABLE movies_data
  FIELDS TERMINATED BY ','
  ENCLOSED BY '\"'
  LINES TERMINATED BY '\n'
  IGNORE 1 ROWS;
  "

dbExecute(con, load_movies_data_query)
```

```
## [1] 60
```

```
load_movies_query <- "
  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/movies.csv'
  INTO TABLE movies
  FIELDS TERMINATED BY ','
  ENCLOSED BY '\"'
  LINES TERMINATED BY '\n'
  IGNORE 1 ROWS;
  "

dbExecute(con, load_movies_query)
```

```
## [1] 6
```

```
load_critics_query <- "
  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/critics.csv'
  INTO TABLE critics
  FIELDS TERMINATED BY ','
  ENCLOSED BY '\"'
  LINES TERMINATED BY '\n'
  IGNORE 1 ROWS;
  "
dbExecute(con, load_critics_query)
```

```
## [1] 44
```

```
load_joint_table_query <- "
  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/joint_table.csv'
  INTO TABLE joint_table
  FIELDS TERMINATED BY ','
  ENCLOSED BY '\"'
  LINES TERMINATED BY '\n'
  IGNORE 1 ROWS;
  "

dbExecute(con, load_joint_table_query)
```

```
## [1] 60
```

```
dbDisconnect(con)
```

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