Movie Ratings Analysis

MD SIMON CHOWDHURY

2024-09-15

Introduction

In this week, we gather movie ratings from several individuals, store them in a normalized MySQL database, transfer the data into R, handle missing data, and explore options for enhancing security and standardizing ratings.

```
# Load required libraries
library(DBI)
library(RMySQL)

# Connect to MySQL database on localhost
conn <- dbConnect(RMySQL::MySQL(),
                  dbname = "movies_poll",
                  host = "localhost",
                  port = 3306,
                  user = "root",
                  password = Sys.getenv("MYSQL_PASSWORD"))

# Query data
movie_data <- dbGetQuery(conn, "SELECT * FROM ratings")
```

```
# View the data
print(movie_data)
```

```
##    rating_id person_id movie_id rating
## 1          1         1        1      5
## 2          2         1        2      4
## 3          3         1        4      3
## 4          4         1        5      5
## 5          5         1        6      4
## 6          6         2        1      4
## 7          7         2        3      5
## 8          8         2        4      4
## 9          9         2        5      3
## 10        10         2        6      2
## 11        11         3        2      5
## 12        12         3        3      4
## 13        13         3        4      4
## 14        14         3        5      5
## 15        15         3        6      5
```

Handling Missing Data

Several ratings are missing (NA values). I decided to impute the missing ratings by filling them with the average rating for that movie. I used the dplyr package to replace missing ratings (NA) with the average rating for the respective movie.

```
# Load necessary library
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Replace NA values with the mean rating for each movie
movie_data_imputed <- movie_data %>%
  group_by(movie_id) %>%
  mutate(rating = ifelse(is.na(rating), mean(rating, na.rm = TRUE), rating)) %>%
  ungroup()

# View imputed data
print(movie_data_imputed)
```

```
## # A tibble: 15 × 4
##    rating_id person_id movie_id rating
##        <int>     <int>    <int>  <int>
##  1         1         1        1      5
##  2         2         1        2      4
##  3         3         1        4      3
##  4         4         1        5      5
##  5         5         1        6      4
##  6         6         2        1      4
##  7         7         2        3      5
##  8         8         2        4      4
##  9         9         2        5      3
## 10        10         2        6      2
## 11        11         3        2      5
## 12        12         3        3      4
## 13        13         3        4      4
## 14        14         3        5      5
## 15        15         3        6      5
```

Standardizing Ratings

To make ratings comparable across users, I standardized the ratings using z-scores. By standardizing the ratings into z-scores, we ensure that each user's ratings are centered around their personal average, making comparisons fairer.

```
# Standardize ratings by calculating z-scores
movie_data_standardized <- movie_data %>%
  group_by(person_id) %>%
  mutate(z_rating = scale(rating, center = TRUE, scale = TRUE)) %>%
  ungroup()

# View standardized ratings
print(movie_data_standardized)
```

```
## # A tibble: 15 × 5
##    rating_id person_id movie_id rating z_rating[,1]
##        <int>     <int>    <int>  <int>        <dbl>
##  1         1         1        1      5        0.956
##  2         2         1        2      4       -0.239
##  3         3         1        4      3       -1.43
##  4         4         1        5      5        0.956
##  5         5         1        6      4       -0.239
##  6         6         2        1      4        0.351
##  7         7         2        3      5        1.23
##  8         8         2        4      4        0.351
##  9         9         2        5      3       -0.526
## 10        10         2        6      2       -1.40
## 11        11         3        2      5        0.730
## 12        12         3        3      4       -1.10
## 13        13         3        4      4       -1.10
## 14        14         3        5      5        0.730
## 15        15         3        6      5        0.730
```

Conclusion

In conclusion, we collected movie ratings from five individuals and stored them in a normalized MySQL database. The ratings were transferred to R, where I handled missing data using mean imputation. The ratings were then standardized using z-scores to allow fair comparisons across different users. By using environment variables to manage sensitive information like database credentials, I improved the security of the database.

The average ratings for each movie were calculated, and the best-rated movie was identified.

```r
# Calculate average ratings for each movie
avg_ratings <- movie_data_imputed %>%
  group_by(movie_id) %>%
  summarise(avg_rating = mean(rating, na.rm = TRUE)) %>%
  arrange(desc(avg_rating))

# Get the movie name for the highest-rated movie
best_movie_id <- avg_ratings$movie_id[1]
best_movie_name <- dbGetQuery(conn, paste("SELECT movie_name FROM movies WHERE movie_id =", b
est_movie_id))

# Display the best-rated movie
cat("The best-rated movie is:", best_movie_name$movie_name, "with an average rating of", avg_
ratings$avg_rating[1], "\n")
```

```
## The best-rated movie is: Oppenheimer with an average rating of 4.5
```

```r
# Close the connection
dbDisconnect(conn)
```

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