DATA 607 Week 2A SQL and R

Benjamin Bravo

2026-02-05

Approach Deliverable

Approach/Introduction

This project will collect and analyze simple movie rating data using a SQL database and R. I will select six recent popular movies and ask at least five people to rate each movie they have seen on a 1–5 scale. Participants will be allowed to skip movies they have not watched, which will result in missing ratings.

The data will be stored in a relational database using a normalized structure with separate tables for users, movies, and ratings. All tables will be created and populated using SQL code.

After the data is stored, I will load it into R as a dataframe by querying the database directly. In R, I will inspect the data, handle missing ratings appropriately, and compute basic summaries such as average ratings and counts per movie and user.

Anticipated Challenges

The main challenge is missing data, since not all participants will rate every movie. This will be handled by allowing NULL values in the database and excluding missing ratings from summary calculations. Another challenge is ensuring correct joins between tables, which will be addressed by carefully validating IDs and relationships.

R

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.2
## v purrr     1.2.0
## -- 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 errors
```

```
data_url <- "https://raw.githubusercontent.com/bb2955/607-week2/main/Week_2_SQL_and_R.csv"

ratings <- readr::read_csv(data_url, show_col_types = FALSE)
```

```r
install.packages("conflicted")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```r
library(conflicted)

conflict_prefer_all("dplyr", quiet = TRUE)
```

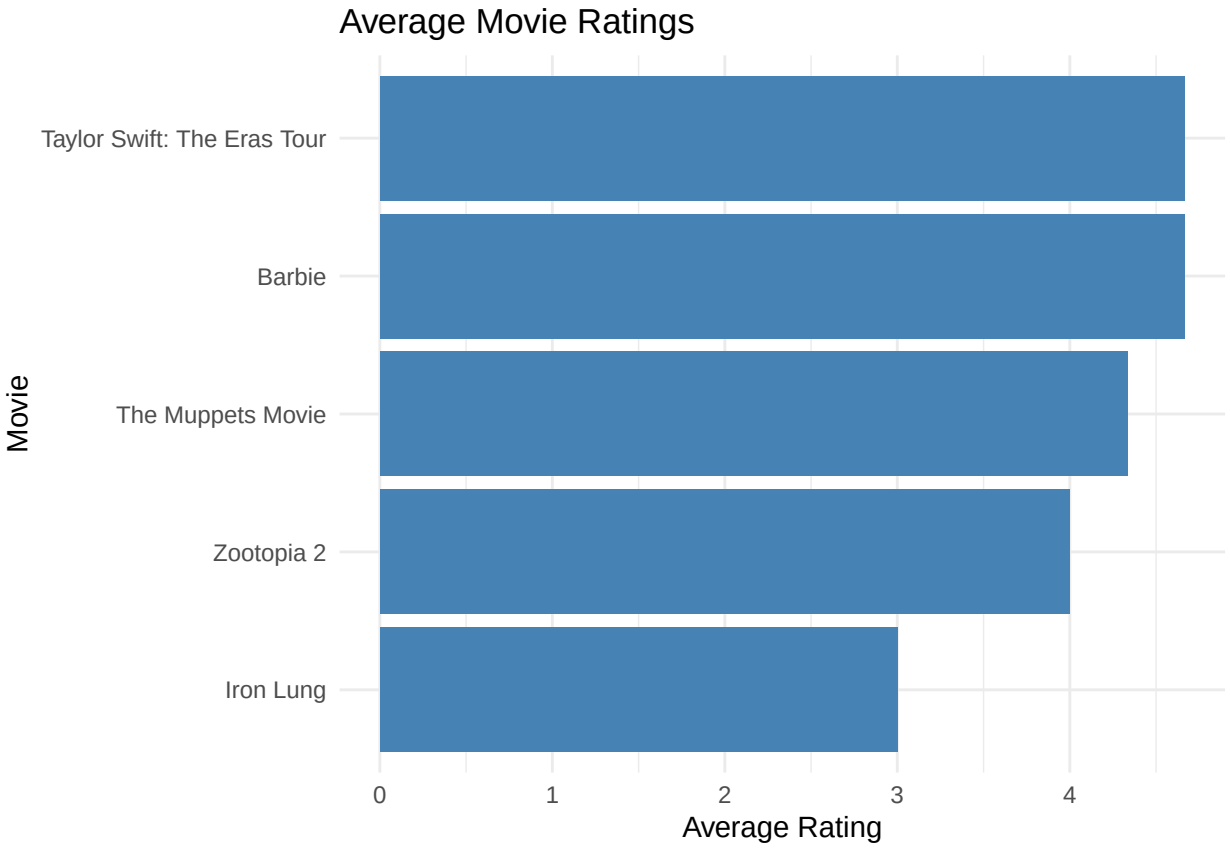
```r
glimpse(ratings)
```

```
## Rows: 13
## Columns: 3
## $ Name   <chr> "Alex", "Alex", "Alex", "Jordan", "Jordan", "Jordan", "Sam", "S~
## $ Title  <chr> "Zootopia 2", "The Muppets Movie", "Barbie", "Zootopia 2", "Iro~
## $ Rating <dbl> 5, 4, 5, 4, 3, 5, 5, 4, 3, 4, 4, 5, 5
```

```r
ratings %>%
  group_by(Title) %>%
  summarise(
    avg_rating = mean(Rating, na.rm = TRUE),
    num_ratings = n()
  )
```

```
## # A tibble: 5 x 3
##   Title                       avg_rating num_ratings
##   <chr>                            <dbl>       <int>
## 1 Barbie                            4.67           3
## 2 Iron Lung                         3              1
## 3 Taylor Swift: The Eras Tour       4.67           3
## 4 The Muppets Movie                 4.33           3
## 5 Zootopia 2                        4              3
```

```r
ratings %>%
  group_by(Title) %>%
  summarise(avg_rating = mean(Rating, na.rm = TRUE)) %>%
  ggplot(aes(x = reorder(Title, avg_rating), y = avg_rating)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Average Movie Ratings",
    x = "Movie",
    y = "Average Rating"
  ) +
  theme_minimal()
```



Conclusion

This assignment demonstrated how to collect user ratings, store them in a relational SQL database, and analyze the data in R. Using a normalized database structure made it easy to manage missing values and summarize movie ratings. This work could be extended by collecting more ratings, visualizing results, or exploring simple recommendation techniques.

SQL Code Used:

CREATE TABLE users (user_id SERIAL PRIMARY KEY, name TEXT);

CREATE TABLE movies (movie_id SERIAL PRIMARY KEY, title TEXT);

CREATE TABLE ratings (user_id INTEGER, movie_id INTEGER, rating INTEGER, FOREIGN KEY (user_id) REFERENCES users(user_id), FOREIGN KEY (movie_id) REFERENCES movies(movie_id));

INSERT INTO users (name) VALUES ('Alex'), ('Jordan'), ('Sam'), ('Taylor'), ('Chris');

INSERT INTO movies (title) VALUES ('Zootopia 2'), ('The Muppets Movie'), ('Iron Lung'), ('Barbie'), ('Taylor Swift: The Eras Tour');

INSERT INTO ratings (user_id, movie_id, rating) VALUES (1, 1, 5), (1, 2, 4), (1, 4, 5), (2, 1, 4), (2, 3, 3), (2, 5, 5), (3, 2, 5), (3, 4, 4), (4, 1, 3), (4, 5, 4), (5, 2, 4), (5, 4, 5), (5, 5, 5);

SELECT u.name, m.title, r.rating FROM ratings r JOIN users u ON r.user_id = u.user_id JOIN movies m ON r.movie_id = m.movie_id ORDER BY u.name;