Data607-Week_Two-Assignment

Shri Tripathi

09/15/2024

Week Two - R and SQL

Overview

In this assignment, I will be working with movie ratings data and using both SQL and R for data management and analysis. First, I'll gather ratings from at least five individuals for six recent popular movies, using a scale of 1 to 5. Next, I'll store this data in a MySQL database hosted on Azure. After that, I'll transfer the data from the SQL database into an R dataframe for further analysis. Finally, I'll implement a strategy to handle any missing data and explain why I chose that approach.

In this survey, 5 people will be asked to rate the following six recent, popular movies on a scale of 1 (Worst) to 5 (Best):

1. **Spider-Man: Across the Spider-Verse**
2. **John Wick: Chapter 4**
3. **Guardians of the Galaxy Vol. 3**
4. **Mission: Impossible – Dead Reckoning Part One**
5. **Oppenheimer**
6. **Murder Mystery 2**

SQL

After gathering the movie ratings data, I created a new schema and database in MySQL called assignment_2. Then, I set up a table to store the data and inserted the movie ratings provided by five participants. Below is the SQL code used for creating the database and inserting the data.

Creating the database and table:

```sql
CREATE DATABASE IF NOT EXISTS assignment_2;

USE assignment_2;

CREATE TABLE IF NOT EXISTS movie_ratings (
  ind_id INT NOT NULL,
  name VARCHAR(45) NOT NULL,
  movie_title VARCHAR(45) NOT NULL,
  movie_rating INT NOT NULL,
  PRIMARY KEY (ind_id, movie_title)
);
```

Adding the data to the table movie_ratings:

```sql
INSERT INTO movie_ratings (ind_id, name, movie_title, movie_rating)
VALUES
       (1, 'john', 'spider_man_across_the_spider_verse', 5),
       (1, 'john', 'john_wick_chapter_4', 4),
       (1, 'john', 'guardians_of_the_galaxy_vol_3', 3),
       (1, 'john', 'mission_impossible_dead_reckoning', 5),
       (1, 'john', 'oppenheimer', 4),
       (1, 'john', 'murder_mystery_2', 3),

       (2, 'sarah', 'spider_man_across_the_spider_verse', 4),
       (2, 'sarah', 'john_wick_chapter_4', 5),
       (2, 'sarah', 'guardians_of_the_galaxy_vol_3', 4),
       (2, 'sarah', 'mission_impossible_dead_reckoning', 4),
       (2, 'sarah', 'oppenheimer', 5),
       (2, 'sarah', 'murder_mystery_2', 4),

       (3, 'mark', 'spider_man_across_the_spider_verse', 3),
       (3, 'mark', 'john_wick_chapter_4', 5),
       (3, 'mark', 'guardians_of_the_galaxy_vol_3', 5),
       (3, 'mark', 'mission_impossible_dead_reckoning', 4),
       (3, 'mark', 'oppenheimer', 3),
       (3, 'mark', 'murder_mystery_2', 2),

       (4, 'emma', 'spider_man_across_the_spider_verse', 5),
       (4, 'emma', 'john_wick_chapter_4', 4),
       (4, 'emma', 'guardians_of_the_galaxy_vol_3', 4),
       (4, 'emma', 'mission_impossible_dead_reckoning', 5),
       (4, 'emma', 'oppenheimer', 4),
       (4, 'emma', 'murder_mystery_2', 3),

       (5, 'david', 'spider_man_across_the_spider_verse', 2),
       (5, 'david', 'john_wick_chapter_4', 3),
       (5, 'david', 'guardians_of_the_galaxy_vol_3', 4),
       (5, 'david', 'mission_impossible_dead_reckoning', 4),
       (5, 'david', 'oppenheimer', 5),
       (5, 'david', 'murder_mystery_2', 3);
```

Note: The above code may not run directly in an R Markdown file as I haven't yet configured the RMD file to connect to MySQL on localhost. However, this SQL script will work in MySQL Workbench or any SQL client connected to the database.

Importing to R

Now, I will import the data from the MySQL table into R for further use. To achieve this, I utilized the `DBI` and `RMySQL` libraries. I established a connection to MySQL on my localhost, provided my credentials, and retrieved the data as a dataframe.

Note: This code may not run directly in the R Markdown file as I haven't yet figured out how to include the connection without exposing my password. However, if you want to run this code, ensure that you first execute the SQL code above on your own SQL server, then update the code below with your SQL credentials. I'll continue to learn how to securely handle credentials in future implementations.

```r
# Load the necessary libraries
library(DBI)
library(RMySQL)

# Connect to the MySQL database
connect = dbConnect(RMySQL::MySQL(),
                 dbname = "assignment_2",
                 host = "localhost",
                 user = "your_username",    # Replace with your MySQL username
                 password = "your_password")  # Replace with your MySQL password

# SQL query to retrieve the data from the movie_ratings table
query = "SELECT * FROM movie_ratings"

# Store the result of the query in a dataframe
movie_ratings = dbGetQuery(connect, query)

# Display the first few rows of the dataframe
head(movie_ratings)
```

Discussion

Handling Missing Data There are various ways to handle missing data, and one approach is to create new data points from the existing ones. After importing the data from SQL, I created a new dataframe that contains the median rating for each movie. I achieved this by aggregating the ratings for each movie, calculating the median, and then visualizing the results using `ggplot`. This method helps in dealing with missing or incomplete data by relying on the central tendency of the available ratings, providing a balanced representation for each movie.

R code to calculate and visualize the median ratings for each movie:

```r
# Split function to extract the ratings of each movie
list_of_ratings = split(movie_ratings$rating, movie_ratings$movie_title)

# Display the list of ratings by movie
list_of_ratings

# Calculate the median rating for each movie
median_ratings = aggregate(rating ~ movie_title, data = movie_ratings, FUN = median)

# Display the new data frame with median ratings
median_ratings

# Load ggplot2 for visualization
library(ggplot2)

# Create the bar plot to visualize the median ratings by movie
ggplot(median_ratings, aes(x = movie_title, y = rating)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +  # Rotate x-axis labels for readability
  labs(title = "Median Ratings by Movie",
       x = "Movie Title",
       y = "Median Rating")
```

Another way I added missing data from the existing dataset was by creating a key that assigned a genre to each movie and then incorporating this genre data into the dataframe. After that, I visualized each individual's movie genre preferences.

Adding genres to each movie:

```r
# Creating a genre key for each movie
genre_key = c("fantasy", "interactive", "sci_fi", "comedy", "drama", "comedy")
names(genre_key) = median_ratings$movie_title

# Convert the genre key into a data frame for merging
genre_df = data.frame(movie_title = names(genre_key), genre = genre_key)

# Merging the genre data with the movie_ratings data frame
movie_ratings = merge(movie_ratings, genre_df, by = "movie_title")

# Display the updated data frame
head(movie_ratings)
unique(movie_ratings$movie_title)
```

Visualizing the rating distribution by genre:

```r
# Using ggplot to plot the rating distribution across genres
ggplot(movie_ratings, aes(x = genre, y = rating, color = ind_name)) +
  geom_jitter(width = 0.3, height = 0) +
  labs(title = "Distribution of Ratings by Genre",
       x = "Genre",
       y = "Rating") +
  theme(legend.position = "bottom")
```

Aggregating each person's movie genre preferences using dplyr:

```r
# Loading the dplyr library for data manipulation
library(dplyr)

# Grouping the data by individual person and genre, then summarizing the average rating per genre
aggregated_ratings = movie_ratings %>%
  group_by(ind_name, genre) %>%
  summarise(avg_rating = mean(rating)) %>%
  ungroup()

# Display the aggregated data
aggregated_ratings
```

Heatmap of average ratings by genre per individual:

```r
# Creating a heatmap to visualize average ratings by genre for each individual
ggplot(aggregated_ratings, aes(x = ind_name, y = genre)) +
  geom_tile(aes(fill = avg_rating), color = "white") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Average Ratings Heatmap",
       x = "Individual",
       y = "Genre")
```

This approach adds meaningful information (genres) to the dataset and enables visual exploration of individual preferences, allowing for deeper insights into the data. The heatmap and jitter plot help illustrate how each person rates different genres, offering a personalized view of their movie preferences.

Another important consideration from the assignment is the possibility that not all participants watched every movie. I didn't offer an "N/A" option for movies they hadn't seen. If I had included this option, I could have handled missing data by removing any NULL entries, depending on factors like sample size and context. There are multiple ways to address missing data. To improve this assignment in the future, I would include an "N/A" option and use tools like Google Forms, Microsoft Forms, or RedCap to collect data. Additionally, I would increase the sample size and work on ensuring my code runs securely without exposing my MySQL password.

Bonus Challenge Questions:

You're encouraged to optionally find other ways to make your solution better. For example, consider incorporating one or more of the following suggestions into your solution:

Use survey software to gather the information.

- Tools like Google Forms, Microsoft Forms, or RedCap could streamline data collection and ensure consistency. They would also allow for the inclusion of an "N/A" option for unwatched movies.

Are you able to use a password without having to share the password with people who are viewing your code? • There are a lot of interesting approaches that you can uncover with a little bit of research.

- Currently, I've hardcoded my MySQL password in the R code, which isn't secure. A better approach would be to use **environment variables** or **config files** to securely store sensitive information. In R, I could use the `dotenv` package or store credentials in a `.Renviron` file. Another option would be to implement key management services like **AWS Secrets Manager** or **Azure Key Vault**, which would allow me to retrieve the password securely without exposing it directly in the code.

While it's acceptable to create a single SQL table, can you create a normalized set of tables that corresponds to the relationship between your movie viewing friends and the movies being rated?

- Currently, I have a single table (movie_ratings) that combines movies, participants, and ratings. To normalize my database:
- I could create a separate table for users (e.g., user_data with columns user_id and user_name).
- Then, create a table for movies (e.g., movie_data with columns movie_id, movie_title, and genre).
- Finally, I could have a ratings table that uses foreign keys to link user_id and movie_id to record the corresponding rating. This would reduce data redundancy and make the database more flexible for future additions.

Example Schema:

```sql
CREATE TABLE user_data (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(50)
);

CREATE TABLE movie_data (
    movie_id INT PRIMARY KEY,
    movie_title VARCHAR(50),
    genre VARCHAR(50)
);

CREATE TABLE ratings (
    user_id INT,
    movie_id INT,
    rating INT,
    FOREIGN KEY (user_id) REFERENCES user_data(user_id),
    FOREIGN KEY (movie_id) REFERENCES movie_data(movie_id)
);
```

Is there any benefit in standardizing ratings? How might you approach this?

Standardizing ratings could be useful to compare ratings across individuals who may have different scales or biases in their ratings. For example, one person's "4" might be another person's "5." You could standardize the ratings by calculating z-scores or normalizing the data on a 0-1 scale. This would allow for a more objective comparison of movie ratings across different users.

```
movie_ratings$z_score_rating <- scale(movie_ratings$rating)
```