Movie Ratings Sample Data

John Cruz

2023-02-02

Introduction

I will be collecting movie ratings through an online survey on a scale of 1 to 5.

The overall steps of this project is the following:

1. Collect survey data
2. Create a database to store this information
3. Ability to query the data into R
4. Analyze some descriptive measures.

Collecting Survey Data

The survey was conducted using a Google Forms named *Spoiled Corn*. This survey will collect ratings of six recent popular movies on a scale of 1 to 5, 1 being the lowest. If a person did not watch this movie, it will be categorized with an option of N/A. Once data has been collected, I exported the data via CSV, so that it can be imported into a PostgreSQL database.

Online Survey

Building the Database

The database used was PostgreSQL for no particular reason but to learn how to use another database option outside of my previous ones such as SQL Server, duckDB and Google Cloud Storage.

1. Using PostgreSQL within pgAdmin 4, create a database named *movie_survey*.
2. Within the *movie_survey* database, run the following code: create_tables.sql

- The SQL code is created based off this ER Diagram.
- I created a table named *surveys* to store the data in its original format. After this initial table is made, I used the following code allowing the CSV file to be imported to the *surveys* table.

```
copy surveys from 'CSV survey filepath' delimiter ',' HEADER csv;
```

- To create the *films* table, I used the column names from the *surveys* table to name each row and also grab its release year.

```
SELECT
    SPLIT_PART(column_name, '(', 1) AS movie_name,
    SPLIT_PART(SPLIT_PART(column_name, '(', 2), ')', 1) AS release_year
FROM information_schema.columns
WHERE table_schema = 'public'
    AND table_name   = 'surveys'
```

- The *ratings* table was created in the same format as the *films* table, however, because I wanted to unpivot the columns of the *surveys* table, similar to using the *melt()* function in Pandas and I was having issues with R's version, I was able to use two functions, *LATERAL()* and *VALUES()*, to perform the same idea.

```
SELECT s."Timestamp", s.user_id, val.*
FROM surveys s,
LATERAL(
    VALUES
        (1, s."Avengers: Endgame (2019)"),
        (2, s."The Lion King (2019)"),
        (3, s."Joker (2019)"),
        (4, s."Everything Everywhere All at Once (2022)"),
        (5, s."The Matrix Resurrections (2021)"),
        (6, s."King Richard (2021)")
    ) AS val (movie_id, full_rating)
```

- I queried the database to see if it works as intended.

Connecting to our Database

Required Libraries

```
library('RPostgres')
library('DBI')
library('tidyverse')
```

PostgreSQL Database

Using this connection we will be able to keep our username and password from being embedded into our code and instead have RStudio pop up a dialog box to input the information.

```
cnxn <- dbConnect(RPostgres::Postgres(),
                  dbname = 'movie_survey',
                  port = 5432,
                  user = 'postgres',
                  password = 'elliebelly968')
                  # user = rstudioapi::askForPassword("Database Username"),
                  # password = rstudioapi::askForPassword("Database Password"))
```

Query the Database

Here we are able to directly query our PostgreSQL database and pull the information from the tables we created earlier. *dbFectch()* will allow us to move the results of the query into a readable DataFrame.

```
query <- dbSendQuery(cnxn,
                     "SELECT
                          r.datetimeid,
                          r.user_id,
                          f.name,
                          r.rating,
                          f.release_year
                      FROM ratings r
                      INNER JOIN films f
                          ON r.movie_id = f.movie_id ")

df <- dbFetch(query)
```

Close Connection

Since we have gathered the data we needed from the database, we will close the connection to PostgreSQL.

```
dbDisconnect(cnxn)
```

```
## Warning in connection_release(conn@ptr): There is a result object still in use.
## The connection will be automatically released when it is closed
```
