SQL Assignment Part 1

Devin Cline

2023-03-13

This assignment is worth 100 points. There are 15 questions (1a, 1b, 1c, 2a, etc…) and each question is worth 6.5 points each. This assignment is due Tuesday March 14 by 11:59pm CST.

Once you have completed the assignment, render the RMarkdown document to a pdf. Save the pdf as 'SQL_Assignment_Part1)your_last_name.' Upload the pdf to the SQL Assignment Part 1 link in Module 4 in Canvas.

Remember, the file will not render if the View() function is not either removed or commented out or if there are any errors in the code chunks.

Do not forget to call the sqldf library each time you invoke a new R session (that is, you exit and re-enter RStudio Cloud). The sqldf package is already installed in this project.

```r
#call the sqldf package
library(sqldf)
```

Data Sets and Scenario

You are going to use three data tables in this assignment. These data tables are related to information about films. The name of the tables are films, reviews, and people.

Run the following code chunk below that will create the three tables in your R Studio environment.

```r
#create data table films
films <- read.csv("/cloud/project/data/films.csv")

#create the data table people
people <- read.csv("/cloud/project/data/people.csv")

#create the table reviews
reviews <- read.csv("/cloud/project/data/reviews.csv")
```

Time to Query!

1a. Count the number of records in the people table, aliasing the result as count_records.

Hint: The count should be 8397.

```r
sqldf("SELECT COUNT(*) AS count_records
       FROM people")
```

```
##   count_records
## 1          8397
```

1b. Count the number of languages and countries in the films data table. Use aliases count_languages and count_countries, respectively.

Hint: You can use multiple COUNT() statements in a SELECT function. There should be 48 languages and 65 countries.

```
sqldf("SELECT COUNT(DISTINCT language) AS count_languages,
      COUNT(DISTINCT country) AS count_countries
      FROM films")
```

```
##   count_languages count_countries
## 1              48              65
```

1c. Select the title of every film that doesn't have a budget associated with it and use the alias no_budget_info. Use the films data table. Limit the report to 10 rows. You can run the code chunk below to see what the first 10 rows should look like.

```
sqldf("SELECT title AS no_budget_info
       FROM films
       WHERE budget IS NULL LIMIT 10")
```

```
##                        no_budget_info
## 1                         Pandora's Box
## 2                 The Prisoner of Zenda
## 3                         The Blue Bird
## 4                                 Bambi
## 5                            State Fair
## 6                           Open Secret
## 7                     Deadline - U.S.A.
## 8                                 Ordet
## 9                      The Party's Over
## 10 The Torture Chamber of Dr. Sadism
```

2a. Generate a report that has film_id and facebook_likes with less than 1,000 likes from the reviews table. Limit the report to 10 rows. You can run the code chunk below to see what the first 10 rows should look like.

```
sqldf("SELECT film_id, facebook_likes
       FROM reviews
       WHERE facebook_likes < 1000 LIMIT 10")
```

```
##    film_id facebook_likes
## 1      285              0
## 2       65            491
## 3       83            930
## 4      111              0
## 5       59            689
## 6      163              0
## 7      402              0
## 8      251            912
## 9      113            872
## 10     107            975
```

2b. Count how many records have a num_votes of at least 100,000 from the reviews table; use the alias films_over_100K_votes. Hint: The count should be 1211.

```
sqldf("SELECT COUNT(*) AS films_over_100k_votes
       FROM reviews
       WHERE num_votes > 100000")
```

```
##   films_over_100k_votes
## 1                  1211
```

2c. Select and count the language column using the alias count_spanish from the films data table. Apply a filter to select only Spanish from the language field.

Hint: There should 40 films in Spanish.

```
sqldf("SELECT language, COUNT(language) AS count_spanish
       FROM films
       WHERE language = 'Spanish'")
```

```
##   language count_spanish
## 1  Spanish            40
```

3. Create a report that meets the following requirements.

a. Count the unique titles from the films data table and use the alias nineties_english_films_for_teens.
b. Filter to include only movies with a release_year from 1990 to 1999, inclusive.
c. Add another filter narrowing your query down to English-language films.
d. Add a final filter to select only films with 'G', 'PG', 'PG-13' certifications.

Hint: The count should be 310.

```
sqldf("SELECT COUNT(DISTINCT title) AS ninteties_english_films_for_teens
       FROM films
       WHERE release_year BETWEEN 1990 AND 1999
       AND language = 'English'
       AND certification IN ('G', 'PG', 'PG-13')")
```

```
##   ninteties_english_films_for_teens
## 1                               310
```

4a. Calculate the average amount grossed by all films whose titles start with the letter 'A' and alias with avg_gross_A. Round the average to zero decimal places.

Hint: The average should be $47,893,236 (no commas and dollar signs will be printed).

```
sqldf("SELECT ROUND(AVG(gross), 2) AS avg_gross_A
       FROM films
       WHERE title LIKE 'A%'")
```

```
##   avg_gross_A
## 1    47893236
```

4b. Select the lowest gross film in 2016 and use the alias lowest_gross. The report should contain the title of the film.

Hint: The name of the film is Operation Chromite and the gross amount is $31,662 (no commas or dollar signs will be printed).

```
sqldf("SELECT title, MIN(gross) AS lowest_gross
       FROM films
       WHERE release_year = 2016")
```

```
##                title lowest_gross
## 1 Operation Chromite        31662
```

5a. Select the title and duration from every film, from longest duration to shortest. Use the films data table. Limit the report to 10 rows. You can run the code chunk below to see what the first 10 rows should look like.

```
sqldf("SELECT title, durations
       FROM films
       ORDER BY durations DESC LIMIT 10")
```

```
##                        title durations
## 1                     Carlos       334
## 2        Blood In, Blood Out       330
## 3              Heaven's Gate       325
## 4  The Legend of Suriyothai       300
## 5                   Das Boot       293
## 6             Apocalypse Now       289
## 7                The Company       286
## 8         Gods and Generals       280
## 9                 Gettysburg       271
## 10  Arn: The Knight Templar       270
```

5b. The ORDER BY can be used to sort multiple columns. It will sort by the first column specified, then sort by the next, and so on. Columns on the ORDER BY function should be separated by commas with no comma after the last column.

Create a SQL query that will generate a report exactly as shown in the code chunk below. Limit the report to 10 rows.

```
sqldf("SELECT title, release_year, durations
       FROM films
       WHERE release_year IS NOT NULL AND durations IS NOT NULL
       ORDER BY release_year, durations DESC LIMIT 10")
```

```
##                                               title release_year durations
## 1  Intolerance: Love's Struggle Throughout the Ages         1916       123
## 2                       Over the Hill to the Poorhouse         1920       110
## 3                                     The Big Parade         1925       151
## 4                                         Metropolis         1927       145
## 5                                       Pandora's Box         1929       110
## 6                                 The Broadway Melody         1929       100
## 7                                      Hell's Angels         1930        96
## 8                                 A Farewell to Arms         1932        79
## 9                                        42nd Street         1933        89
## 10                                  She Done Him Wrong         1933        66
```

6a.Select the release_year and count of films released in each year aliased as film_count. Do not include records where release_year is missing. Use the films data table.Limit the report to 10 rows. You can run the code chunk below to see what the first 10 rows should look like.

```
sqldf("SELECT release_year, COUNT(title) AS film_count
       FROM films
       WHERE release_year IS NOT NULL
       GROUP BY release_year LIMIT 10")
```

```
##    release_year film_count
## 1          1916          1
## 2          1920          1
## 3          1925          1
## 4          1927          1
## 5          1929          2
## 6          1930          1
## 7          1932          1
## 8          1933          2
## 9          1934          1
## 10         1935          1
```

6b. GROUP BY becomes more useful when it is used on multiple columns or in conjunction with ORDER BY. You apply GROUP BY to multiple columns in the same manner you do ORDER BY (multiple columns separated by commas with no comma after the last column).

You want to understand budget changes throughout the years in individual countries.

Select the release_year, country, and the maximum budget aliased as max_budget for each year and each country. Sort the report by release_year and country. You do not want the report to include missing values for release_year or budget. Limit the report to 10 rows. You can run the code chunk below to see what the first 10 rows should look like.

```
sqldf("SELECT release_year, country, MAX(budget) AS max_budget
       FROM films
       WHERE release_year IS NOT NULL AND budget IS NOT NULL
       GROUP BY country, release_year
       ORDER BY release_year, country LIMIT 10")
```

```
##    release_year country max_budget
## 1          1916     USA     385907
## 2          1920     USA     100000
## 3          1925     USA     245000
## 4          1927 Germany    6000000
## 5          1929     USA     379000
## 6          1930     USA    3950000
## 7          1932     USA     800000
## 8          1933     USA     439000
## 9          1934     USA     325000
## 10         1935     USA     609000
```

7. Combining filtering and sorting provides you greater interpretability by ordering reports. You are interested in what countries have the highest average film budgets.

Create a report that meets the following requirements. a. Select the country and the average budget as average_budget, from the films table. b. Group the results by country. c. Filter the results to countries with an average budget of more than one billion (1000000000). d. Sort by descending order of the average_budget.

You can run the code chunk below to see what report should look like.

```
sqldf("SELECT country, AVG(budget) AS average_budget
       FROM films
       GROUP BY country
       HAVING average_budget > 1000000000
       ORDER BY average_budget DESC")
```

```
##       country average_budget
## 1 South Korea     1383960000
## 2     Hungary     1260000000
```

8. Generate a report that that returns the average budget and gross earnings for films each year after 1990 if the average gross budget is greater than 60 million.

This query is a real-world scenario. Many times, you will be asked to write a more complex query that answers a specific business question that cannot be found by playing around in applications like Excel.

To provide you some assistance, you can run the code chunk below to see what the final report should look like.

```
sqldf("SELECT country, release_year, AVG(budget) as avg_budget, AVG(gross) as avg_gross
       FROM films
       WHERE release_year > 1990
```

```
      GROUP BY release_year
      HAVING avg_gross > 60000000
      ORDER BY avg_budget DESC")
```

```
##   country release_year avg_budget avg_gross
## 1     USA         2016   56642742  76924036
## 2     USA         2012   41331819  62873528
## 3     USA         2015   39298329  72573303
## 4     USA         2014   35325799  62412137
## 5     USA         1992   25982030  63665195
```