Data 607 - Assignment 2

Richie R.

Overview

In this assignment, we will be taking data that I've collected from a few friends and uploading it to a SQL table. To make it harder for myself, I'm using GCP BigQuery for my table. I've chosen BigQuery because it has a relatively generous free tier. I am also deciding to normalize my data a little bit.

First we will import the libraries we need. We are assuming that these are installed.

```
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
```

```
library(magrittr)

library(bigrquery)

project_id <- "cuny-msds"
dataset_name <- "data607"

path_to_bq_creds <- "F:/git/cuny-msds/data607/gcp_bq_auth/gcp_bq_auth.json"
```

1. Documenting the movie reviews

In this step, I will create an R list of lists which contains the individual's name and their reviews. Each list will contain the below parameters:

1. name
2. movie
3. rating

To make the code a bit more readable, I'm going to create variables for each movie

```r
mv_pt <- "Poor Things"
mv_op <- "Oppenheimer"
mv_bb <- "Barbie"
mv_mi <- "Mission: Impossible - Dead Reckoning Part One"
mv_sm <- "Spider-Man: Across the Spider-Verse"

movie_ratings <- list(
  list(name = "Mehru", rating = 5, movie = mv_pt),
  list(name = "Mehru", rating = 4, movie = mv_op),
  list(name = "Mehru", rating = 5, movie = mv_bb),
  list(name = "Mehru", rating = 3, movie = mv_mi),
  list(name = "Mehru", rating = 5, movie = mv_sm),
  list(name = "Adrian", rating = 5, movie = mv_op),
  list(name = "Adrian", rating = 5, movie = mv_bb),
  list(name = "Adrian", rating = 5, movie = mv_sm),
  list(name = "Lucas", rating = 5, movie = mv_sm),
  list(name = "Sheethal", rating = 4, movie = mv_op),
  list(name = "Sheethal", rating = 4, movie = mv_bb),
  list(name = "Sheethal", rating = 4, movie = mv_sm),
  list(name = "Megan", rating = 1, movie = mv_bb),
  list(name = "Megan", rating = 1, movie = mv_sm)
)
```

2. Converting the data into a dataframe

With this list of lists, we'll convert it into a dataframe by applying as.data.frame to each element in the list.

```r
ratings_df <- do.call(rbind, lapply(movie_ratings, as.data.frame))
head(ratings_df)
```

```
##     name rating                                        movie
## 1  Mehru      5                                  Poor Things
## 2  Mehru      4                                  Oppenheimer
## 3  Mehru      5                                       Barbie
## 4  Mehru      3 Mission: Impossible - Dead Reckoning Part One
## 5  Mehru      5           Spider-Man: Across the Spider-Verse
## 6 Adrian      5                                  Oppenheimer
```

3. Normalizing the datasets

To normalize this data, we'll create 3 tables:

1. users - consisting of a user_id and a user_name
2. movies - consisting of a movie_id and a move_name
3. ratings - consisting of a rating_id and a rating_name

Using the dataframe, let's assume that it contains all of the ratings that need to be loaded.

```r
users_df <- ratings_df %>%
  distinct(name) %>%
  mutate(user_id = row_number())

ratings_df <- subset(
  merge(ratings_df, users_df, by = "name")
  , select = -name
)

movies_df <- ratings_df %>%
  distinct(movie)       %>%
  mutate(movie_id = row_number())

ratings_df <- subset(
  merge(ratings_df, movies_df, by = "movie")
  , select = -movie
)
```

4. Loading the data into BigQuery

First we will need to authenticate:

```r
bigrquery::bq_auth(path = path_to_bq_creds)
```

```
## ! Using an auto-discovered, cached token.

##   To suppress this message, modify your code or options to clearly consent to
##   the use of a cached token.

##   See gargle's "Non-interactive auth" vignette for more details:

##   <https://gargle.r-lib.org/articles/non-interactive-auth.html>

## i The bigrquery package is using a cached token for 'rtk0211@gmail.com'.
```

Now that we've authenticated, we will upload the users table

```r
users_table <- bq_table(
  project = project_id,
  dataset = dataset_name,
  table = "users"
)

bq_table_create(
  x = users_table,
  fields = as_bq_fields(users_df)
)
```

```
## <bq_table> cuny-msds.data607.users
```

```
bq_table_upload(
  x = users_table,
  values = users_df,
  write_disposition = "WRITE_TRUNCATE"
)
```

Moving onto the movies table:

```
movies_table <- bq_table(
  project = project_id,
  dataset = dataset_name,
  table = "movies"
)

bq_table_create(
  x = movies_table,
  fields = as_bq_fields(movies_df)
)
```

```
## <bq_table> cuny-msds.data607.movies
```

```
bq_table_upload(
  x = movies_table,
  values = movies_df,
  write_disposition = "WRITE_TRUNCATE"
)
```

Finally we'll write the movie_ratings dataset:

```
movie_ratings_table <- bq_table(
  project = project_id,
  dataset = dataset_name,
  table = "movie_ratings"
)

bq_table_create(
  x = movie_ratings_table,
  fields = as_bq_fields(ratings_df)
)
```

```
## <bq_table> cuny-msds.data607.movie_ratings
```

```
bq_table_upload(
  x = movie_ratings_table,
  values = ratings_df,
  write_disposition = "WRITE_TRUNCATE"
)
```

Conclusion

With the data collected from our 5 wonderful volunteers, we were able to normalize the data into reference tables and also use the bigrquery package to upload it to BigQuery. This was my first time using R to do so making this something new for me!