Homework 6

For this homework you will create an R Markdown document that outputs to a PDF and you will upload both the .Rmd and .pdf files to wolfware. Be sure to put your name in the title of the document.

The purpose of this homework is to get practice reading data from APIs and databases. It is a bit more open ended than previous homework assignments.

To Do - Write a markdown file that goes through each step below. For problems requiring an explanation, just answer with text.

Databases

BigQuery

1. We'll start by connecting to a database - Google's BigQuery database (sandbox anyway). We can connect to this for free (and without putting in a credit card) by going to (https://cloud.google.com/bigquery/docs/sandbox)[https://cloud.google.com/bigquery/docs/sandbox].

Follow the steps on there to create an account and a project. You should never have to put in a credit card for the sandbox. If you do and somehow gets charged Justin Post and NC State are in no way liable for the costs you may incur by doing things outside the realm of this assignment (there that makes me feel better).

Now open up R and connect to the database. You'll need to put your project id in here:

```r
# install devtools if you haven't already
# devtools::install_github('r-dbi/bigrquery')
library(bigrquery)
library(DBI)
con <- dbConnect(bigrquery::bigquery(), project = "publicdata", dataset = "samples",
    billing = "your_project_id")
```

We can see what tables are available for us to use with `dbListTables()` (you'll likely be prompted to verify your credentials via a message in the R console - which means you need to run this in console prior to knitting).

```r
dbListTables(con)
```

Now you can practice running SQL queries and queries through dplyr - although things were updated recently so I'm having trouble getting functions like `select()` to work correctly. Place a two queries that you did below (say one SQL query using `dbGetQuery` and one `dplyr` queries using `dplyr` functions - check out the help for `dbGetQuery` for an example using SQL). In SQL code, `LIMIT 10` will only grab 10 results (I'd add that to your query so you don't have to wait long for the results).

Local Database

We will be working with an example database called chinook. It has tables of songs, playlists, customers, invoices, and more. See the 'chinook-database-diagram.pdf' to get a better understanding of how these tables are related to one another.

1) Download the 'chinook.db' database. Install and load the `DBI` and `RSQLite` packages, and load the `tidyverse` package as well. Then use `dbConnect` and `RSQLite::SQLite` to connect to the chinook

database, store it as an object called `db`. Then print out the tables in db using `dbListTables`. (This is done for you below.)

```
library(DBI)
library(RSQLite)
library(tidyverse)
db <- dbConnect(RSQLite::SQLite(), "chinook/chinook.db")
dbListTables(db)
```

2) Query the db to return to "playlists" and "playlist_track" tables (just run `tbl(db, "playlists")`. Store them as 'pl' and 'pl_tracks' respectively. How many playlists are in this database?

3) Use an `inner_join` to combine pl and pl_tracks by the 'PlaylistId' column, call this 'pl_join'. Print out a frequency table to show how many tracks are in each playlist by their title by using `table(pl_join$Name)`.

4) Read in the "tracks" table and store it as an object called 'tracks' and the "albums" table as 'albums'.

5) Filter `pl_join` to just the "Grunge" playlist, and call this object grunge. Combine grunge and 'tracks' together with a left join, and print the output to the console. What does this look like, and why?

6) Use the rename function to change 'grunge$Name' to 'PlaylistName' and 'tracks$Name' to 'TrackName'. Now retry the join, and store it as 'grunge_tracks'. This should look better!

7) It is good practice to disconnect from a database that you are accessing over the web or a server when you are done querying it. Even though ours is just a local file, we can still do that. Use `dbDisconnect` to disconnect from the database.

Querying an API

For this section, you may want to check out this blog post about pulling data from any API into R. https://www.programmableweb.com/news/how-to-access-any-restful-api-using-r-language/how-to/2017/07/21

There are a few common steps to accessing data from any API with R.

- Install/load the "httr" and "jsonlite" packages
- Create an API account, get a username and password if necessary
- Make a "get" request to the API to pull raw data into your environment
- "Parse" that data from its raw form through JavaScript Object Notification (JSON) into a usable format
- Proceed with data cleaning, manipulation, and analysis

1) Install and load the `httr` and `jsonlite` packages.

Pokemon API (no key)

2) Go to https://pokeapi.co/docs/v2 and look at the documentation for this API. What are some of the endpoints (tables) that we can access from it?

Extract information about Pikachu

3) To access this API, we need to construct a URL with the name of the table and attributes we want to pull from it. Start with a base_URL of https://pokeapi.co/api/v2. We want to pull the "pikachu" attribute from the "pokemon" table.

Paste the pieces together like so:

```
full_url <- paste0(base_url, "/", tab_name, "/", poke_name)
```

4) Use a GET command to retrieve the pikachu information in raw form. Then use '`content`' to turn it into JSON text form. Finally, convert it to a list using '`fromJSON`' with '`flatten=TRUE`'. You should end up with a list of length 17.

5) According to this database, how many moves can Pikachu learn?

6) According to this database, which games/versions is Pikachu available in?

You are now ready to submit to wolfware. Nice!