Project Objectives

This is a solo project where you will write functions to query an API and summarize the data that is returned. You'll put this into a shiny app to allow the user to access the API and summarize the data they request!

Setting Things Up: Creating the Repo

The first step is to create a github repo. All project work should be done within this repo so we can track your activity. You should have at least five substantial commits on the repo or else you will lose credit.

We won't go through the usual process of creating a webpage for this repo.

Instead, we will enable the running of your app from GitHub using `shiny::runGitHub()`. This allows someone to submit one line of code to run a shiny app from files you have available on github!

You should check that this works with an "empty" version of RStudio (that is, one that doesn't have objects you've already created existing in your environment - empty your environment!). You want to make sure anyone can run the app using just the code on the repo (and having the appropriate R packages on their computer)!

Your `README.md` file should have the following things (you can edit this file in RStudio or the github web editor):

- Brief description of the app and its purpose.
- A list of packages needed to run the app.
- A line of code that would install all the packages used (so we can easily grab that and run it prior to running your app).
- The `shiny::runGitHub()` code that we can copy and paste into RStudio to run your app.

You do not need to use github pages for this project (unless you want to).

Find Data You Are Interested In

For this project I'm going to let you choose your own API to query (other than a COVID related API - I don't want you to use an API with data related to COVID - I've graded too many of those in my time). A list of free APIs is given here: https://github.com/public-apis/public-apis (but feel free to use another source if you find one)

Many of these don't require a key. If you are worried about sharing a key on github, use one of the key less ones. For what we are doing it shouldn't matter (as long as you don't sign up for a pay API).

You are going to build your own functions for querying the API (not use a package that does it for you). **Please check the required components for the summarizations below before choosing your API and endpoints to return.** You must be returning at least some data that will work for the required summarizations. (You do not need to create functionality to contact all parts of the API!)

- You should write function(s) to contact your chosen API and return **well-formatted, parsed data in the form of data frame(s)/tibble(s)**.
  - Your function(s) should allow the user to **easily** customize their query to return specific data from the API.
  - You do not need to allow the user to query all parts of the API. Your functions should allow the user at request at least six different types of data. This can come in the form of

    * querying different API endpoints (different parts of the API). For example, this NHL example, there is a function that queries the 'franchise-season-records' API and another that queries the 'franchise-goalie-records'. This is an example of querying different endpoints. (**You can't use the NHL API by the way.**)
    * modifications of a particular endpoint. For the NHL example, you can modify the 'franchiseId' on the 'franchise-goalie-records' endpoint. This is an example of a modification of an endpoint.
    * For this part, you might have two different endpoints. You allow the user to modify three things on the first endpoint (counts as four total) and one thing on the second endpoint (counts as two things)

Know How to Summarize the Data

- Once you have the functions to query the data, you'll want to produce common numerical and graphical summaries (these may change depending on which data you have returned). Some requirements are below:
    - You should create some contingency tables
    - You should create numerical summaries for some quantitative variables at each setting of some of your categorical variables
    - You should create at least four plots utilizing coloring, grouping, etc. All plots should have nice labels and titles.
        * At least one plot that you create should be a plot that we didn't cover in class (say a heatmap or something like that - depends on your data - lots of good examples to consider here https://exts.ggplot2.tidyverse.org/gallery/)

App Requirements

Ok, now you're going to put what you did there into a shiny app! Here are some requirements:

- Your app should have multiple pages (tabs) to it. I don't care if you use the built in tabs for shiny or a package like `shinydashboard` - use the method you prefer. Tab information:
- An `About` tab. The tab should
    - Describe the purpose of the app
    - Briefly discuss the data and its source - providing a link to more information about the data
    - Tell the user the purpose of each tab (page) of the app
    - Include a picture related to the data (for instance, if the data was about the world wildlife fund, you might include a picture of their logo)
- A `Data Download` tab. The tab should
    - Allow the user to specify changes to your API querying functions and return data.
    - Display the returned data
    - Subset this data set (rows and columns)
    - Save the (possibly subsetted) data as a file (`.csv` is fine but whatever you'd like)
- A `Data Exploration` tab. The tab should
    - Allow the user to choose variables/combinations of variables that are summarized via numerical and graphical summaries

    * The user should be able to change the type of plot shown and type of summary reported (this may correspond to different types of data that are downloaded)
    * The user should be able to do some kind of faceting on a plot

- You should have at least two dynamic UI elements!

Submission

You should simply post your github repo URL as your submission.

Notes

- There are some nice shiny additions such as a status bar that you can add to display to the user when your API query is running.
- `aes_string()` is useful when trying to use columns selected by the user in a `ggplot()` plot since the columns are often returned as a string
- Similarly, `!!sym(input$...)` can also be useful to deal with this type of issue generally
- `get()` comes in handy some times as well
- This project is pretty open ended! Have fun with it and make something that you can show off to others - Good luck :)

Rubric for Grading (total = 100 points)

Item	Points	Notes
Repo setup/shiny requirements	20	Worth either 0, 5, 10, 15, or 20
About tab	10	Worth either 0, 5, or 10
`Data Download` Tab/API functions	35	Worth either 0, 5, 10, ... 35
`Data Exploration`/Summaries	35	Worth either 0, 5, ..., or 35

Notes on grading:

- For each item in the rubric, your grade will be lowered one level for each each error (syntax, logical, or other) in the code and for each required item that is missing or lacking a description.
- If your work was not completed and documented using your github repo you will lose up to 50 points.
- You should use Good Programming Practices when coding (see wolfware). If you do not follow GPP you can lose up to 50 points on the project.