PA 3 Task Card – Finding Margaret's College

Today you will use the `dplyr` package to clean some data. We will then use that cleaned data to figure out what college Margaret has been accepted to.

This task is complex. It requires many different types of abilities. Everyone will be good at some of these abilities but nobody will be good at all of them. In order to solve this puzzle, you will need to use the skills of each member of your group.

Group Roles

When you are the **Developer**, you will type into the Quarto document in RStudio. However, you **do not** type your own ideas. Instead, you type what the Coder tells you to type. You are permitted to ask the Coder clarifying questions, and, if both of you have a question, you are permitted to ask the professor. You are expected to run the code provided by the Coder and, if necessary, to work with the Coder to debug the code. Once the code runs, you are expected to collaborate with the Coder to write code comments that describe the actions taken by your code.

When you are the **Coder**, you are responsible for reading the instructions / prompts and directing the Developer what to type in the Quarto document. You are responsible for managing the resources your group has available to you (e.g., cheatsheet, textbook). If necessary, you should work with the Developer to debug the code you specified. Once the code runs, you are expected to collaborate with the Developer to write code comments that describe the actions taken by your code.

Group Norms

Remember, your group is expected to adhere to the following norms:

1. Think and work together. Do not divide the work.
2. You are smarter together.
3. Be open minded.
4. No cross-talk with other groups.
5. Communicate with each other!

Dealing with Errors

As you work through this PA, you will encounter some code that does not work as you want it to. Don't despair! **Errors** (when R is unable to run your code) and **bugs** (when the code doesn't do what you hoped) are a natural part of coding. Even the best of the best deal with these problems regularly - learning to track down the issue is a skill that you can learn and practice.

Errors can be sneaky - check results *often*!

If a chunk of code runs smoothly without giving you an error or warning this **does not** necessarily mean it accomplished the desired task.

It is a good habit to check the results of your code **every time** you finish a task. In general, I would recommend completing the following tasks **every time** you write a code chunk.

- Include a comment at the beginning of the code chunk that briefly states the purpose of the chunk. Comments in the code chunks come after `#` signs. These comments will remind later readers—which might be your future self!—what the desired output of the code chunk is.
- If you created a new object, take a look at it! You can inspect the object by either clicking its name in your *Environment* tab or by typing its name into the console. Make sure it looks about how you expect. **Do not** type code to inspect the object in your Quarto file, as that **is not** code that needs to be saved!
- If you created or updated a data frame, make sure your edits did what you hoped. Use the *Environment* or the `head()` function to investigate your changes.

Code Formatting

Don't forget, writing "tidy" and "well documented" code are two of the learning targets for this course. As such, I would strongly encourage you to use every opportunity to practice these skills.

As you are writing code for this assignment, make sure your code follows the tidyverse style guide for dplyr code. Specifically, your code should:

- use whitespace liberally
  - before & after every = sign
  - after every `,`
  - before every `|>`
- use new lines liberally
  - after every `|>`
  - after `,` when needed (if code is more than 80 characters in length)