Lab 02

YOUR NAME HERE

Goal for this lab:

Familiarity with many common data cleaning tasks.

Helpful resources

1. **stringr documentation**- for editing and manipulating strings
2. **dplyr documentation** - for altering datasets, e.g. filterout out rows, adding new columns, grouping and aggregating, etc.
3. **PingPong LLM** (also linked to on the course homepage) - for help when you get stuck - though you can always ask Everett or Huong.

Q0: Download the data, and upload it to DataHub *This is not a coding task. Go to this link to find an download two datasets:*

1. *The first tab is a (scrambled) version of your survey responses. Download this sheet to a csv file, rename it `class_survey_raw.csv`, and upload it into the `data/` folder on DataHub here (look for the import icon in the Files pane).*
2. *The second tab is (fake) "roster" data, like I might get as an instructor from CalCentral. Download the sheet in the second tab to a csv file, name it `roster.csv`, and upload it into the `data/` folder here.*

Load libraries in a single cell at the top, always

Q1: *Make a code cell here and load the `tidyverse` library*

YOUR CODE HERE

Load data

Q2: *Make a cell and load the dataset in `data/class_survey_raw.csv` into a dataframe named `students`*

YOUR CODE HERE

NOTE: *This data (like the data from Lab01) has been shuffled around to anonymize everyone. Each row is a little different from students' actual answers.*

Explore

Q3: *Open the dataset to view it (either click on it in the Environment pane, or run the command `View(students)`. Make a list of at least 5 "issues" that you see need addressing. Use a markdown bulleted list:*

YOUR LIST HERE (not code - just markdown) – hint: either use the word-processor-like buttons to make a list, or see the Markdown cheatsheet for how to do it explicitly.

Clean

Q4: *Get rid of the timestamp column, as it's not really useful to us here. When you're done, look at your dataset to confirm that the column is gone.*

YOUR CODE HERE

Q5: *Rename all the remaining columns to something short, simple, and **lowercase** (e.g. "duck" for the first column, or whatever makes sense to you). Use **underscores_** instead of spaces if you need them, e.g. `first_name` instead of `first name`. The `rename` function may be useful. Note: this may be tricky in that the original questions often have spaces in them - use PingPong for help!*

***Bonus challenge:** Don't do these manually one by one! Make a vector with labels and values (e.g. `renames = c("What duck are you" = "duck", ....)` and write a line of code to apply all of those renamings at once (hint: use `rename_with` and probably ask PingPong for help understanding the syntax)*

YOUR CODE HERE

Q6: *The "what year are you in your program?" column has kind of long answers. Change the contents to just keep the first letter, e.g. instead of `3rd year undergrad` it should just say 3. Make this column an integer (hint: look at `mutate`).*

YOUR CODE HERE

Q7: *The "Fake Test Score" column has numbers between 0 and 1, but there are way more digits than we need. Change this column to round off all these values to two decimal places.*

YOUR CODE HERE

Q8: *In reporting what classes they've taken, some people used uppercase, some lowercase, some mixed. Standardize this. Convert all cells in these columns to be ALL UPPERCASE. Hint: Look at the `stringr` documentation (link at the top of this file), particularly the cheat sheet.*

YOUR CODE HERE

Q9:

A. Make a scatterplot of everyone's comfort scores for both coding and math (put one on the x axis, one on the y). However, instead of just a bunch of dots, plot the `identifier` (a letter) of each person instead of a dot where their scores fall. To do this, instead of geom_point use `geom_text` or `geom_label` (try both and pick the one you like better). You'll have to map not only x and y, but the `label` aesthetic as well. To avoid plotting the labels directly on top of each other, we want some jitter (like we did with points yesterday). Ask PingPong how to accomplish this here, because `geom_jitter` won't work (it only does dots).

B. Tweak the style of your labels to be more clear - adjust the font size, boldness, color, or anything that you think helps improve its legibility. Change at least one thing.

YOUR CODE HERE

Q10: Factors and cats!

*"Factors" in R are strings (words, text), except that 1) they can only take one of a few specific values, and 2) they can be "ordered" arbitrarily. While regular R can handle factors, there is a nice library that makes things much cleaner. This library is called `forcats` (as in "**forcat**egories") check out the docs here, particularly the cheat sheet.*

In our dataset, the column about your major is a clear candidate for a factor. First, run the cell below to see what kind of data is in the `major` column (or whatever you renamed that column to). This code is regular old R, nothing to do with `forcats`

```
#class(students$major)
```

It should say "character" meaning that it is a string of characters (letters, numbers, spaces, etc)

A. Use `group_by` and `summarize` to print how many students have each major

YOUR CODE HERE

B. Turn this column into a factor. Use `mutate` and `factor`.

YOUR CODE HERE

Now run the cell below to see what has changed

```
#class(students$major)
```

It should say "factor" now. Check out the "levels" of the factor - the values they can take, and their order. Run this cell.

YOUR CODE HERE

C. Make a simple bar plot showing how many students have each type of major. Be sure to add a title and rename any axis labels (or remove them).

YOUR CODE HERE

D. Hmm that's a lot of majors with just one student. Wouldn't it be nice to roll all of them up into a single "other" major? You can do this with `forcats`, specifically `fct_lump_min` which will lump together all factor levels that don't pass some minimum threshold. This is a surprisingly common task when doing data analysis! Change the major column to lump together all majors that don't have at least 2 students. You'll need `mutate` and `fct_lump_min`. Check out the `forcats`documentation first, then talk to PingPong if you are struggling.

YOUR CODE HERE

E. Now copy/paste your previous plotting code to see what the distribution of majors looks like now

YOUR CODE HERE

Q11: Bad values and NA *Look at the dataframe, at the column with stats courses. When students haven't taken any STAT courses, sometimes they left the field blank, sometimes they wrote "none", sometimes "N/A", etc. Ideally, all of these would be the same. Typically we want `NA` for all missing values (note: `NA` is not a string, but a native R value meaning "this thing is missing or invalid").*

Find all of these values that people typed to mean "nothing," and convert them all to NA. Use `na_if` to do this. You will need a pipeline with one `na_if` operation for each unique thing like "none" or "N/A" or whatever else people typed.

YOUR CODE HERE

Q12: Save your cleaned data! *A. Save this csv file to a NEW csv file named `class_survey_cleaned.csv` (in the same `data/` folder). Use `write_csv`. This is a common project workflow. You keep the raw data around, have one file of cleaning code, and create a cleaned dataset from which you begin doing real analysis in other code files.*

YOUR CODE HERE

B. This is a common project workflow. You keep the raw data around, have one file of cleaning code, and create a cleaned dataset from which you begin doing real analysis in other code files. To mimic how it will look if you were to follow this pattern, and do your analysis starting from the cleaned file, load back the cleaned dataset (`read_csv`) from its file into a new variable called `students_cleaned`. Look at the dataframe and confirm that it is satisfyingly tidy!

YOUR CODE HERE

Q13: Joining multiple datasets

To submit final grades for the course, I need to compute your grades from Gradescope data, and also tie it together with roster data that I get from CalCentral. This means I have to merge two different datasets.

When I download all of your grades from Gradescope (as a `csv`), there is one row for each of you. It has columns for your name, email, and each assignment.

Separately, I can download the roster data from CalCentral (also a `csv`), which has your student ID, year in your program, etc. I need to somehow get all the information from both of these datasets into one super-dataset, with one row per student. The challenge is that the datasets may be a little different – some students missing from one or the other, "name" columns may be formatted differently so it's hard to automatically know which rows match up, etc. Below, we'll mimic this task, which is VERY common in data science projects (all of science projects, for that matter).

A. The `class_survey_raw` data you have already loaded into `students` and been working with. Now load the other dataset, `calcentral.csv` (which is similar to the roster data I get from CalCentral) into a dataframe called `calcentral`.

YOUR CODE HERE

B. Look at the two datasets. We need some way to know which row from the first dataset "matches" with which row from the second dataset. What common column do the two datasets have? Are there any students who are in one dataset but not the other? Which ones?

YOUR ANSWER HERE

*C. Join the datasets using that shared column, so that I have one row per student, and columns for all the information in both datasets. Save the joined dataset into a variable called `joined`. **Use `outer_join` here**, which will keep all the students, even if they are only in one dataset or the other (and it will fill in some NA values for the data they are missing).*

YOUR CODE HERE

D. Look at the new dataframe you just created. Note the missing values from the students you flagged earlier, the ones who were missing from one or the other dataset. How many rows does this joined dataset have? How many rows do each of the original datasets have?

YOUR ANSWER HERE (you can use code or just read from the environment)

E. Now use `inner_join` to join the two datasets. Same code, just change `outer_join` to `inner_join`

YOUR CODE HERE

F. What's different? How many rows are there? What happened to those students you flagged?

YOUR ANSWER HERE

G. Finally, use `left_join` to join the two datasets. (There is also a `right_join`, and you can guess how that works, but it's equivalent to doing a `left_join` but putting the dataframes in the other order).

YOUR CODE HERE

H. What's different now? How many rows are there? Which students aren't included?

YOUR ANSWER HERE

*I. Given how all of these types of join work, which one do you think I should use in my task of trying to submit final grades for all **enrolled** students?*

YOUR ANSWER HERE