PROJECT 1 COMMENTS

**Project Organization**

- Gather all import libraries in within one code block, with appropriate aliases
- Good understanding of importing/saving using relative paths
- A good executive summary should contain:
    1. Start with an attention-grabbing opening
    2. Define the problem
    3. Describe the solution and expected outcome
    4. Provide evidence that you can deliver
    5. Include a call to action

  https://www.fool.com/the-blueprint/executive-summary/
- Make good use of markdown formatting to organize answers neatly
- Delete unnecessary files if not required, i.e. act_2018.csv
- For multiple code files, use a sequencing prefix such as '1_, 2_, 3_' or 'A_, B_,C_'
- When naming files, use underscore instead of spaces
- If Rubric states clearly Exec Summary in README, it has to be there
- Split the codes into separate notebooks when possible, for neatness

**Clarity of Message**

- A good problem statement should contain:
    - Ideal
    - Reality
    - Consequences
    - Proposal

  https://en.wikipedia.org/wiki/Problem_statement
- Just yes or no is not enough. You need to make your reader understand why your decision is as such.
- Whenever practically possible, link back your observations (data cleaning, EDA, visualization) back to the topic / problem statement / outside research, to expand on why you have observed your data the way it seemed to be, i.e. identifying outliers.

**Python Syntax and Control Flow**

- Respect the Zen of Python http://www.thezenofpython.com/, for example
    - .loc over .iloc
    - Including input names pd.merge(how="outer", on="state")
- Certain functions can generate nicer outputs without using .print()
- Delete unused texts / text blocks / codes / code blocks to declutter the notebook
- Functions should be used to help you to lift off repetitive/complicated tasks.
- Good knowledge of sorting dataframe
- Score for runtime errors by restarting and running notebooks before submission; easy points, get them!
- Python convention is mainly snake_case

**Data Cleaning and EDA**

- Commendable for cleaning the data through Jupyter Notebook, and not manually amending directly on .csv

- Consider memory usage and processing power of your codes, makes a difference with bigger data and heavier workflow, for example
    - int over float
    - .str.lower() with .rename()
- Explicitly show your reader the data has been successfully amended in between the cleaning
- Make a habit to print dataframes within your current screen view using .head() or .tail() instead of the whole dataframe, even when you have a relatively small dataset. Makes your notebook easier to read on GitHub or on Kaggle.
- Applaudable for making extra effort to extract information outside the given .csv files, and double checking them
- Often necessary to do domain research so theoretical minimums and maximums can be used as limits to detect data errors
- Do not assume datasets are fully clean or typo free; future projects might also require checks for duplicate entries
- We might sometimes go back to data cleaning/verification when we visualise outliers later on within a project

**Visualization**

- Make full use of space within the boundary of your notebook and your laptop screen to make the charts bigger and easy to the eyes.
- Customize your plots to show the correct relationship, i.e. putting the same subject side-by-side for better comparison.
- More marks for those who took care in making the labels, legends, axis titles and overall chart more readable and intuitive.
- Heatmaps
    - Always use the triangle masking whenever possible, because it's not meaningful to show the same information twice.
    - Perhaps consider a diverging color scheme, because although it's easy to see the strong correlations (both positive and negatives), it's difficult to differentiate those with weaker correlations.
    - Make color schemes more color blind friendly

**Research and Conceptual Understanding**

- Descriptive and inferential statistics
    - Current dataset contains the means of all US states, not total population of students taking the tests
    - Use charts to help you visualize

**Presentation**

- Be explicit with your problem statement
- Level of technicality is appropriate
- Have a script
- Rehearse!
- Prefer visuals over words and numbers (picture speaks a thousand words)
- Keeping only the key points on slides, and elaborating on the rest

**Overall**

- Regardless of reading the notebook through Jupyter Notebook, GitHub, or Kaggle, restrict it to only scroll vertically. Applies to codes and dataframes.
- Priortizing and time management is important