Remember: tables are just obviously structured data and simple strings-of-texts are just not-so-obviously structured data. First, let us take a look at manipulating tables.

Remember that [terrorism data](https://www.kaggle.com/START-UMD/gtd) which you downloaded last week from Kaggle?

Review the following commands:

csv2tsv < globalterrorismdb\_0617dist.csv | tsv-select -f 2-4,9 > test.tsv

tsv-summarize --header --group-by 4 --count < test.tsv |

tsv-summarize --header --sum 2

tsv-summarize --header --group-by 4 --count ^lt; test.tsv | keep-header --

sort -r -t ' ' -k2 -n | number-lines --header --s "rank" |

tsv-pretty -u | keep-header -- grep "United States"

tsv-summarize --header --group-by 1 --count < test.tsv

NOTES:

1. Each of these is actually a one-line command, but I have inserted line breaks to aid in readability.
2. In the notation *sort -r -t ' '*there is a literal tab character between the single-quotes.

In no more than 500 words, explain what I am doing by each of these commands and why. Also indicate what you would have done differently. Type your response in the text field and submit it to the dropbox.

The first command uses the eBay command *csv2tsv* to convert the Global Terrorism Database (GTB) from comma separated value format to a tab separated format. This serves 2 purposes: first, it makes it significantly easier for a human to read; and second, it allows further processing using the TSV tool. This is then piped into the command *tsv-select*, which pulls out only the columns we are looking for, which in this case are columns 2-4 and 9. By using the “-f” option, we are explicitly looking for fields that match. Finally, we send these selected columns to a file we create called “test.tsv”.

In the second command, we use *tsv-summarize* to get a count of all of the entries in the GTB. The first part of the command groups all of the countries (column 4) into one column and produces a count of each respective country in the second column. We pipe this into another *tsv-summarize* command, and sum all of the individual counts in the second column. In both cases, we use the *–header* option to indicate the first line of the file is a header.

For the second command, it would have been cleaner to use the *wc* command, for instance: *wc -l test.tsv*. However, we would need to know that there was a header, because this command simply gives a count of all of the lines.

In the third command, the first part is identical to the first part of the second command mentioned above. The second part of the command sorts the file while keeping the header out of the sort process. The “-r” flag sorts in reverse order, while the “-t” flag indicates the values are tab separated. The “-k2” flag indicates we are looking at the second column (the counts from above) and the “-n” flag indicates these numeric values are actually stored as strings. Next, this is piped into the *number-lines* command, again keeping the header. Using the “--s” flag, we assign the column the title “rank”. *tsv-pretty* simply outputs the data into a more human readable format, while the “--u” flag underlines the header. Lastly, we use *grep* (again in a header-aware fashion) to pull out the rank and count of the United States.

In this command, there are a few instances that could be optimized. There isn’t any reason to sort the lines in reverse order since we are ultimately only pulling out one line. Also, there is no reason to use *tsv-pretty* mid-command. If the desire is to output it in more human-readable format, this should be moved to the end of the command.

In the fourth command, we are grouping the file by the first column (year) and producing a count of occurrences for each year.