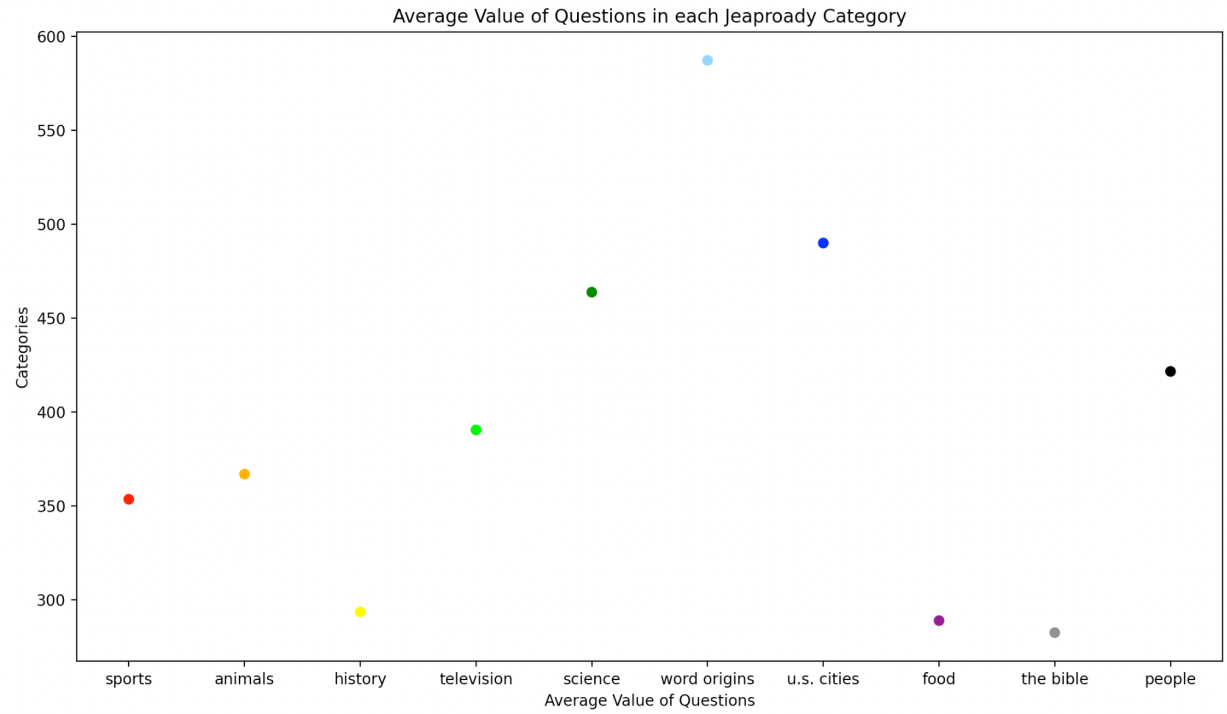
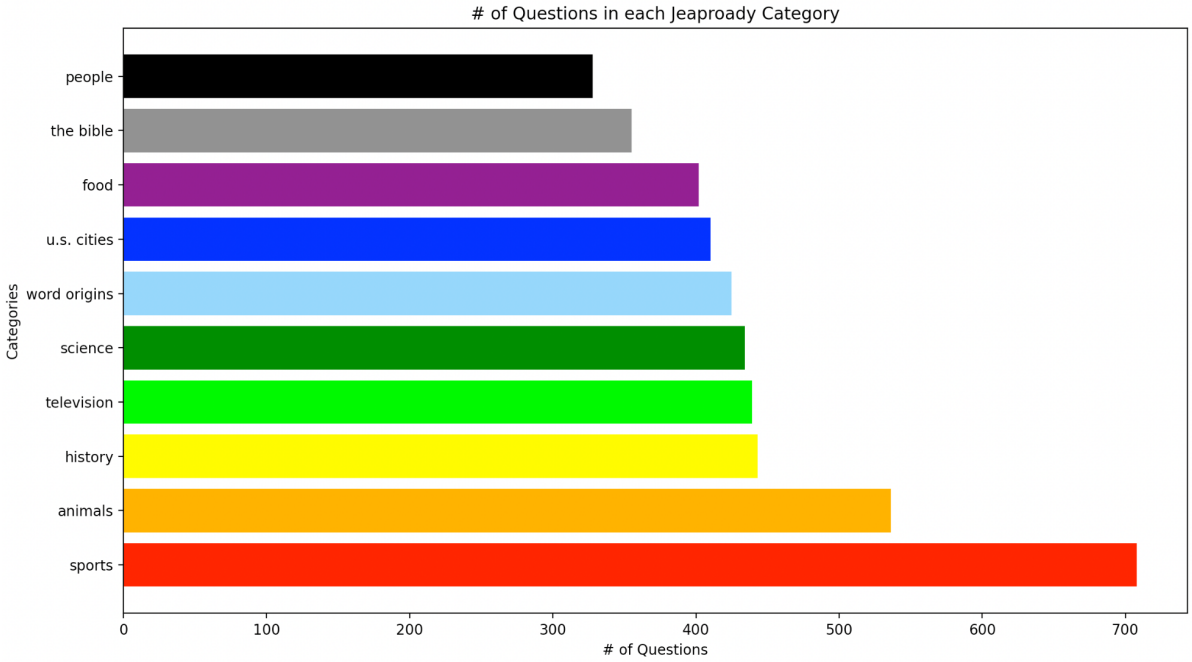
1) The overall goal of this project was to create a tool to help a Jeaproady competitor know what categories are good to study from this API's data set. Specifically, one goal of mine was to display to the user what categories had the most questions to let the user know what categories were historically popular. Another goal of mine was to let the user know the average point value of the questions in each category so that the user knows which categories had the hardest questions. I planned to work with the Jservice API (https://jservice.io/), and I planned to work with specific data about the question count in each category as well as the point values of each question in the dataset.
2) I was able to create a tool that displays which categories were historically popular. In addition, I was able to display the average point values of questions from each category. Because of this, I believe I achieved the goal of creating a tool helpful for Jeaproady competitors to know how to study for the show.
3) One major problem I faced when creating this tool was that the Jservice API had a timeout requirement so that users of the API could not spam and overload the API. However, because of this and how my program was structured (I performed multiple requests to get multiple items to store in my database), if I ran the program too many times in a short period of time, I would reach timeout errors, which significantly slowed down the development of this tool. Another problem I ran into was that the API limited each API result to 100 questions, which made it very inconvenient to find the average point values of all the questions in a category. I was not able to find a good way to find the average value of all the questions in a category, however, I found that finding the average of just a sample of 100 questions from each category had sufficient information to let the user of the tool know the difficulty of the category in the test bank.
4)

```
Category,Question Count,Average Value
sports,708,353.6082474226804
animals,536,367.0
history,443,293.6170212765957
television,439,390.5263157894737
science,434,463.82978723404256
word origins,425,587.2340425531914
u.s. cities,410,490.0
food,402,289.0
the bible,355,282.60869565217394
people,328,421.64948453608247
```

5)





6) Before running the code, you need to install the packages used (pip install __). The packages can be found on the top of the file jp_tool.py. When running this code, all you need to do is type the command “python3 jp_tool.py”. After this, the visualizations from question 5 should pop up. In addition, the calculations can be found in the csv file called calculations.csv. Note that, as mentioned above, that there is a chance of a timeout error. If a timeout error arises, wait 30 seconds to a minute before trying again.

7) Get_questions_from_category takes an input con and cur to make operations on the database and an input of category to know the category ids that have been read and to start reading the category ids that come after the last read category id (sequentially read). There is no explicit output but the function then writes the read in data and puts it into the database. The database stores category id, category name, and question count for each category read in the json. The database also stores question id, question point value, the actual question, and the category id for each question in the json.

Section 1: reads the last read category id from the last run so that the program knows where to start reading for this run. This section also write the last read category id for this run so that the next run knows where to begin reading. This section also calls the get_questions_from_category function.

Section 2: Filter the results to get only the top 10 most popular categories by question count. From the results, the query computes the category, the question count for the category, and the average point value for each category.

Section 3: Writes the data retrieved from section 2 to a CSV file for view.

Section 4: Plots 2 graphs, one that plots category to number of questions in this category as a bar plot and the other plots category to average point value in this category as scatter plot. For both plots, colors added to differentiate between categories. In addition, axis labels and titles are added to these plots.

8)

Date	Issue Description	Location of Resoruce	Result
4/10/23	I was not completely sure how to use the requests library (How to make requests to the API)	https://requests.readthedocs.io/en/latest/	Was able to resolve issue as there was example code to demonstrate how to make a request and turn the result into a JSON
4/11/23	I was not completely sure how to use the API (What API routes to use)	https://jservice.io/	Was able to partially resolve this issue. The documentation on the website gave some help, but was still very unclear. After viewing this site, I still needed to test around and make inferences to make this issue

			completely resolved
4/12/23	I was not completely sure how to read from and write to files	https://www.geeksforgeeks.org/reading-writing-text-files-python/#	Was able to resolve issue as there was example code to demonstrate how to read and write to a file
4/13/23	I did not know how to write a CSV with the calculations	https://docs.python.org/3/library/csv.html	Was able to resolve this issue because there was example code and sufficient documentation to demonstrate how to write a CSV file with headers and specific delimiters
4/14/23	I did not know how to plot a horizontal bar plot and scatter plot with colors	https://matplotlib.org/stable/gallery/color/named_colors.html https://matplotlib.org/stable/gallery/shapes_and_collections/scatter.html#sphx-glr-gallery-shapes-and-collections-scatter-py https://www.python-graph-gallery.com/3-control-color-of-barplots	Was able to resolve this issue because there was example code to show how to plot these graphs with colors. In addition, looked to a site for the strings for specific colors