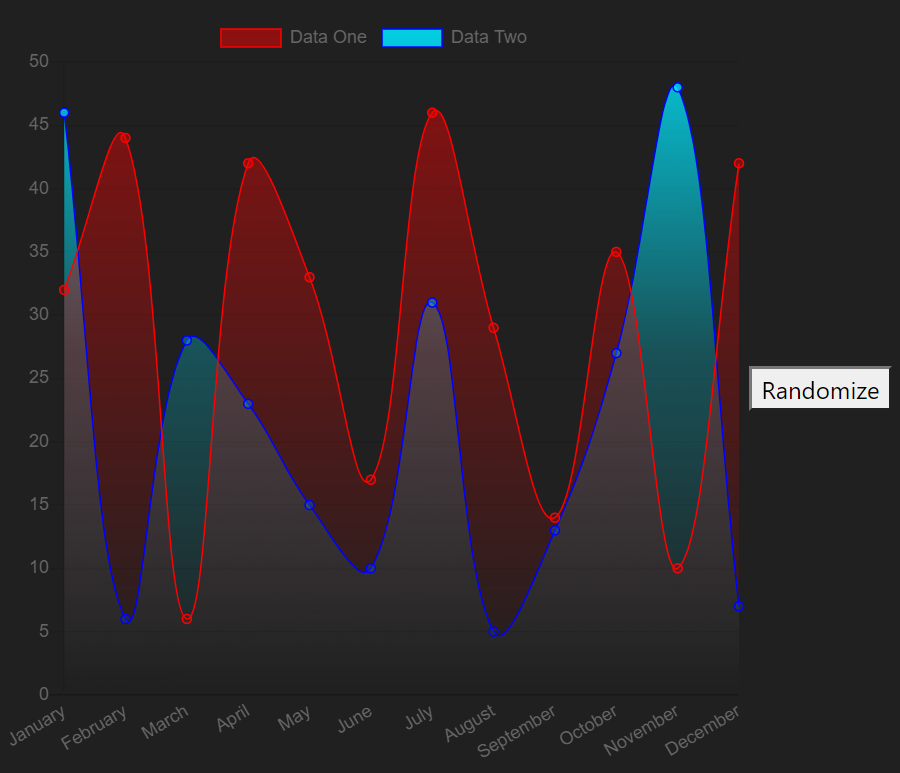
Link to code repo: <https://github.com/nkan001/nic-dsoviz>

The first section of the documentation would be exploratory data visualisations, where I was familiarising myself with the different frameworks and testing out the different possible types of visualizations. The second section would be the main project I worked on, combining my knowledge gathered from section 1.

Section 1 – Exploratory data visualisations

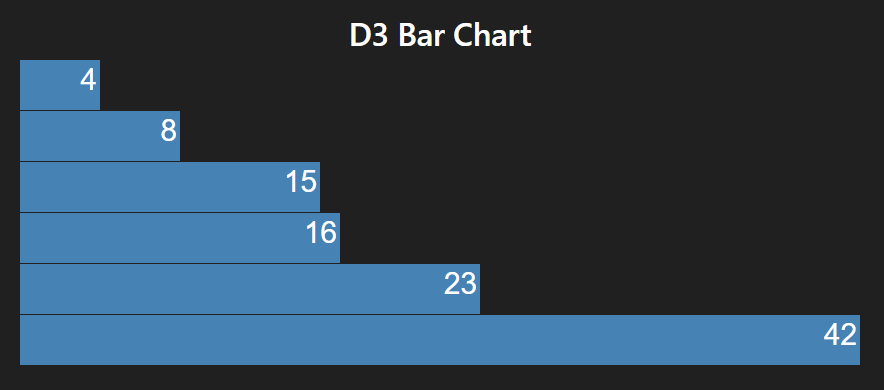
**CHARTS**

1. Gradient Area Chart



* Used Vue-ChartJs to create the area chart
* Created a randomise data function to change values in this.chartdata.
* Added the gradient settings to child component in the mounted lifecycle hook and in the parent component methods. This was done by using this.$refs.canvas to target the chart colour.
* Similar work: charts/bar

2. D3 Bar



* Integrated D3.js into Vue
* Imported the D3 library
* Created the static chart in the mounted lifecycle hook
* Similar works: D3 Star

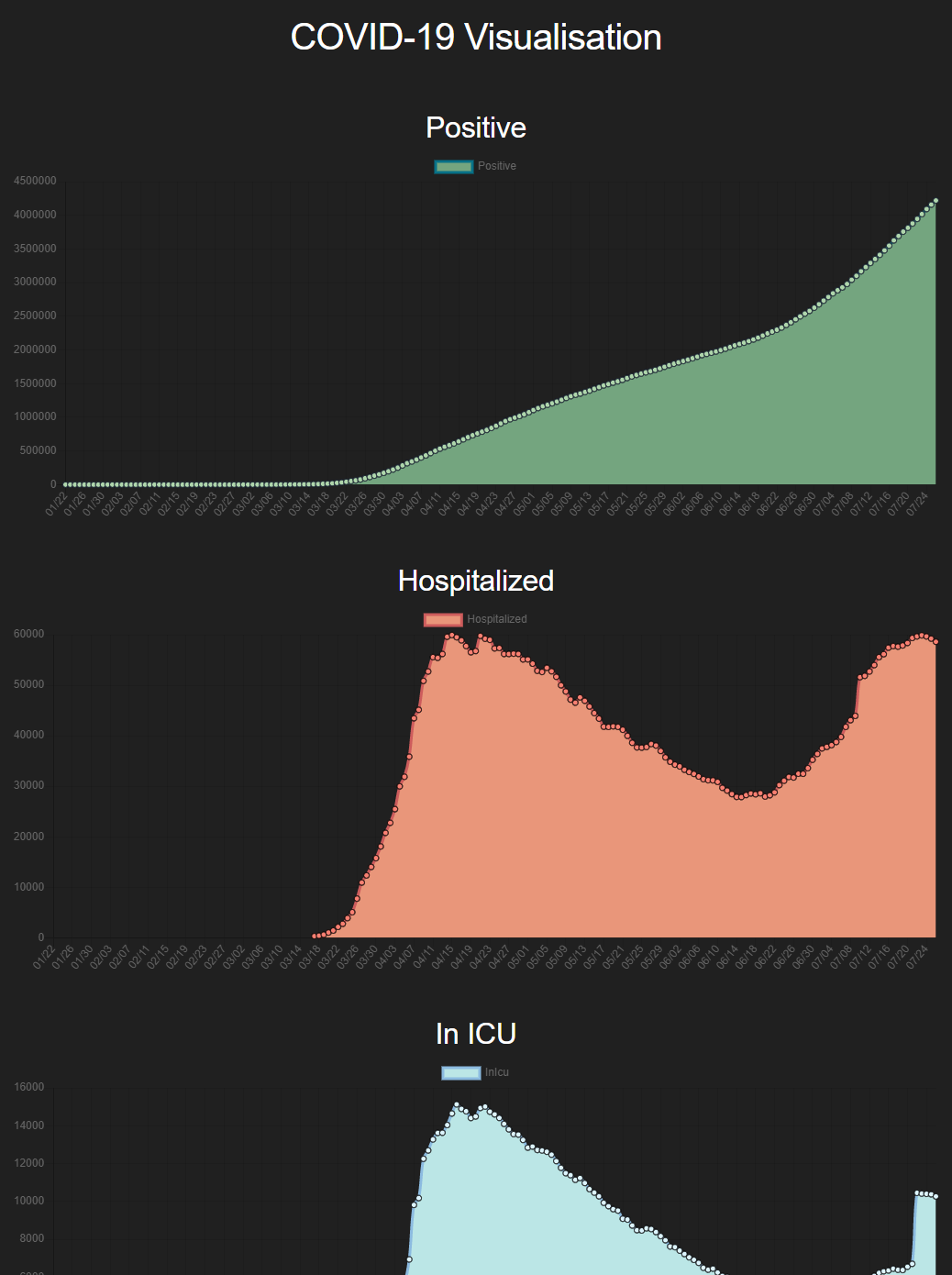
3. D3 Area Chart



* Imported components/d3AreaChart and used props to handle the chart when data changed.
* Gets data to change every 2 seconds
* Uses on mouse over events to show selected value when user hovers over it
* The minimum and maximum value will change respectively depending on user’s input
* Similar work: charts/bubbles

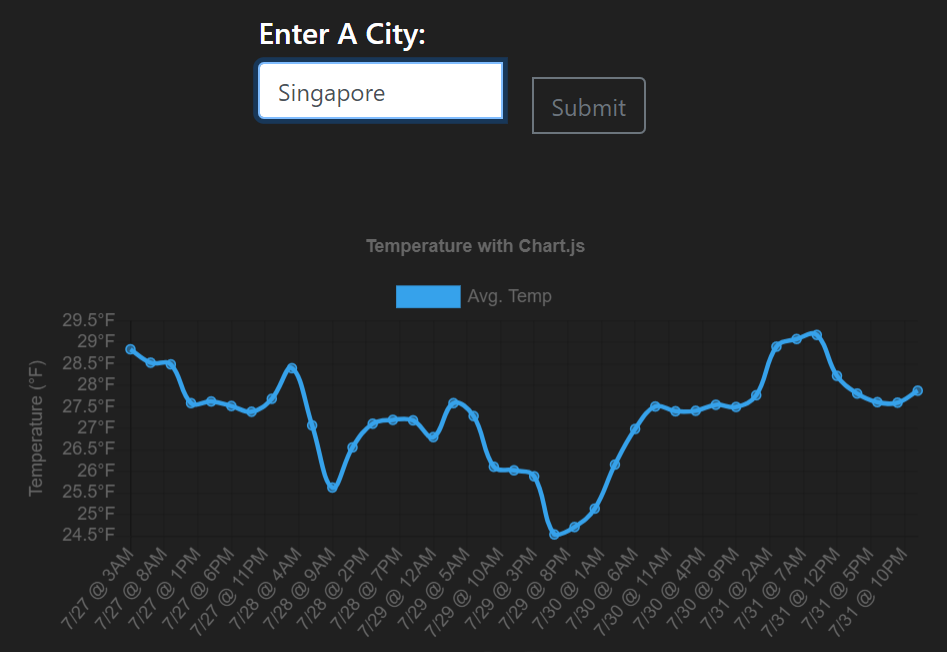
**GRAPHS**

1. COVID-19 US data

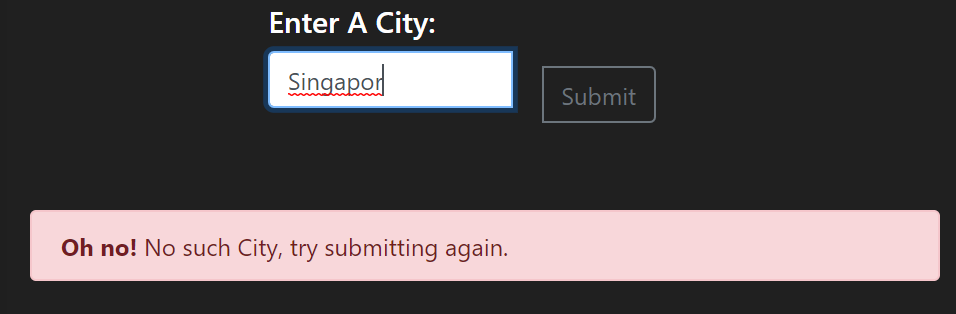


* Used Axios library to fetch API data and async lifecycle hooks to await for the data
* Used Moments library to parse date to the correct format, removing the time and timezone.
* Had Vue-ChartJs Line chart as components

2. Weather Graph

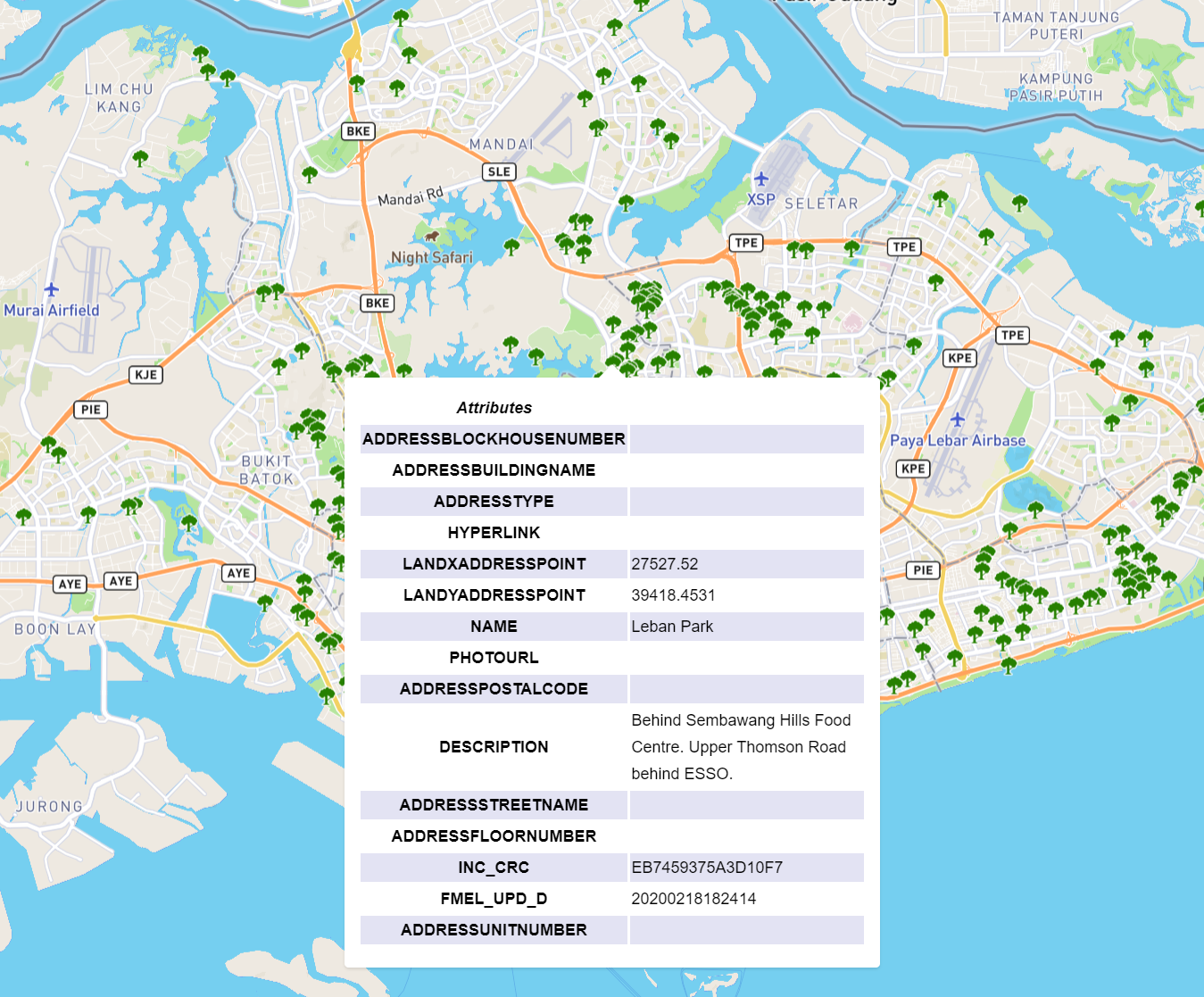


* Allows user to type in any city name. Axios would check it against the OpenWeatherMap API to see if such a city exists, then plots a graph of the temperature over 5 days.
* If the city the user type in is not found in the data online, the page shows:



**GEOSPATIAL**

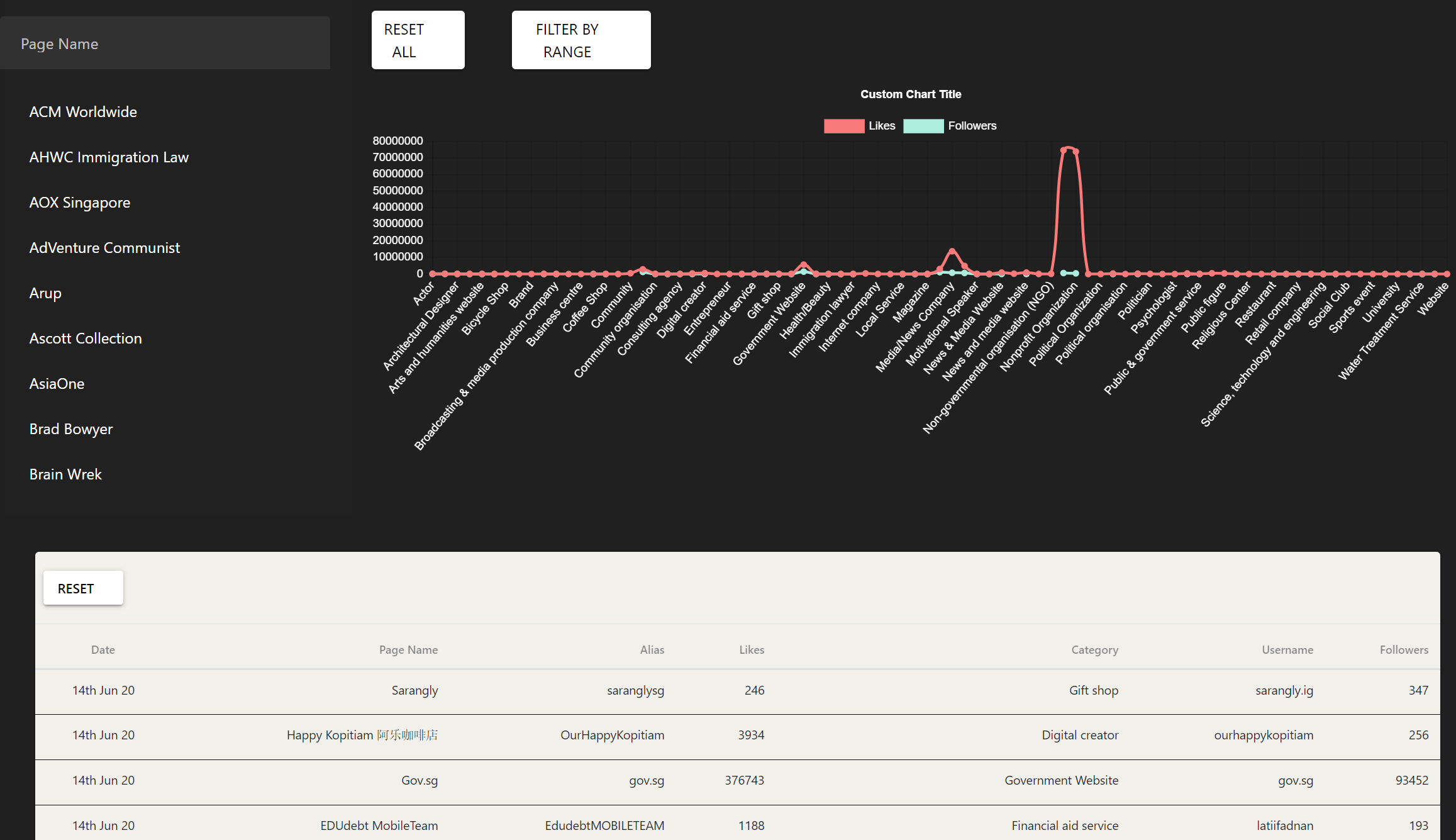
1. Mapbox-gl Parks in Singapore



* Used data.gov.sg data and parsed it as a json object.
* Formatted the data to a presentable format in a table.
* Added tree icons for each area that is a park.
* Used mouseover functions to show the description of places that are hovered over.

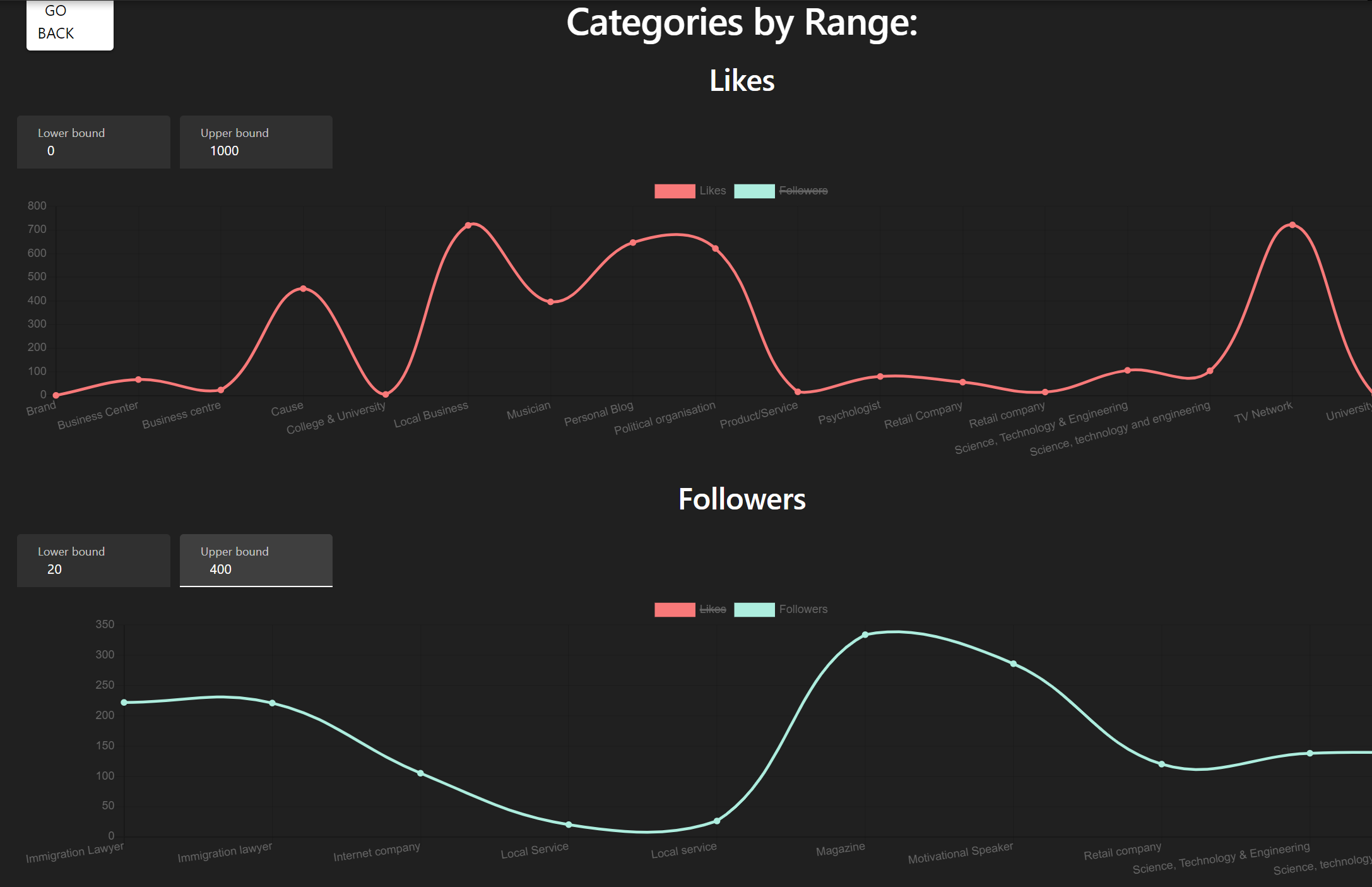
Section 2 – Main work

**FACEBOOK DASHBOARD**



1. Default chart (top right)

* Shows a graph of likes/ followers against the pageCategory. Aggregated all types of categories from the data and summed up the total likes and follower for each category
* Due to some very extreme values, the lower values all look clustered together and indistinguishable. Thus I created a filter by range function, for users to choose the range of values they want to look at.
* The “filter by range button” pushes a new route, but passes on the pageCategory data:



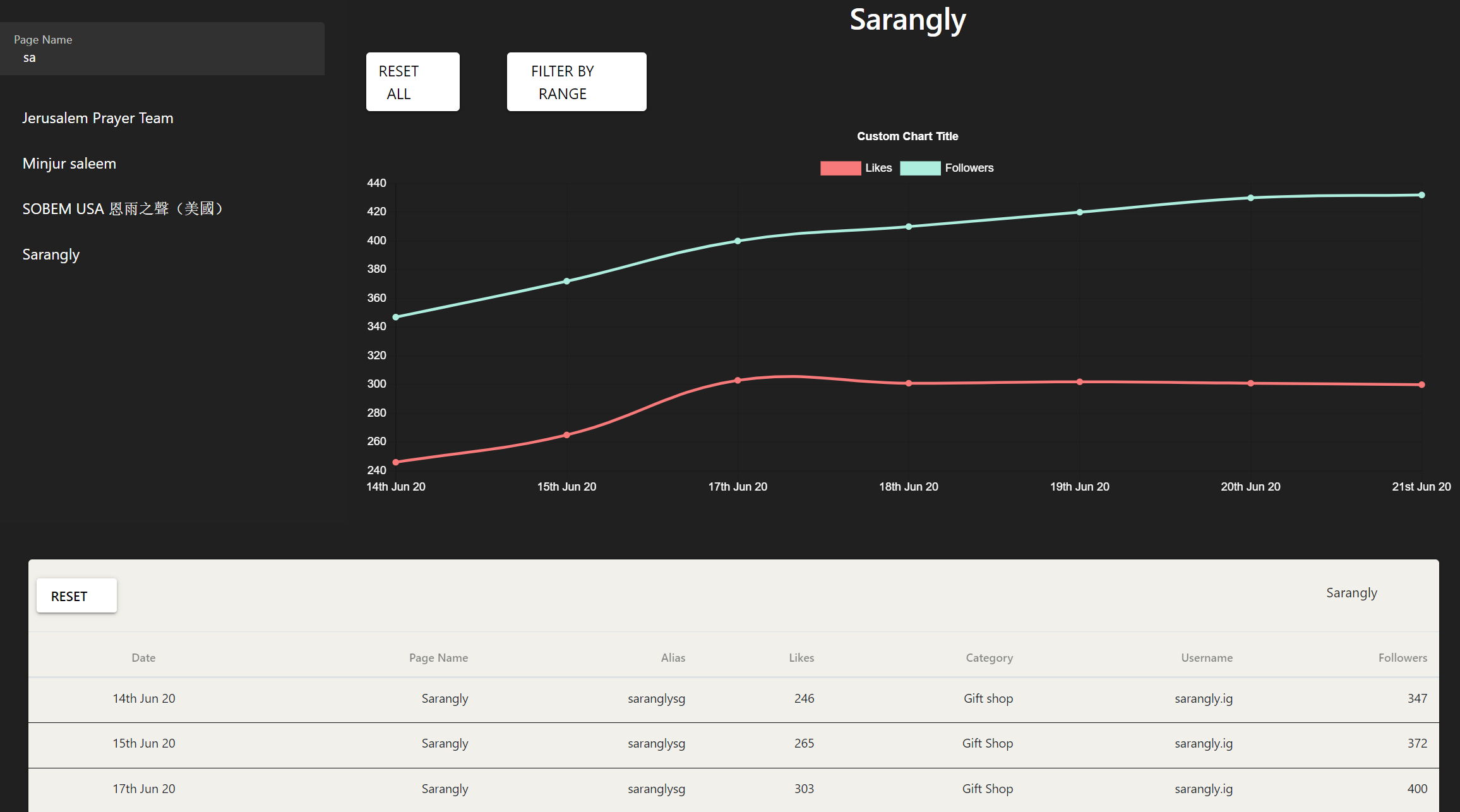
* User can type the lower and upper bounded range they desire in the input boxes above each graph. The graph would change immediately as users type in the numbers. Likes and follower data can be toggled due to the functionality provided for vue-chartjs

2. Table (bottom)

* Displays FB data. I only chose the columns that made more sense and that could be useful for data visualisation.
* Used quasar table, which allows for the number of rows shown to be changed. More data can be seen by moving to the next page. Controls are on the bottom right of the table

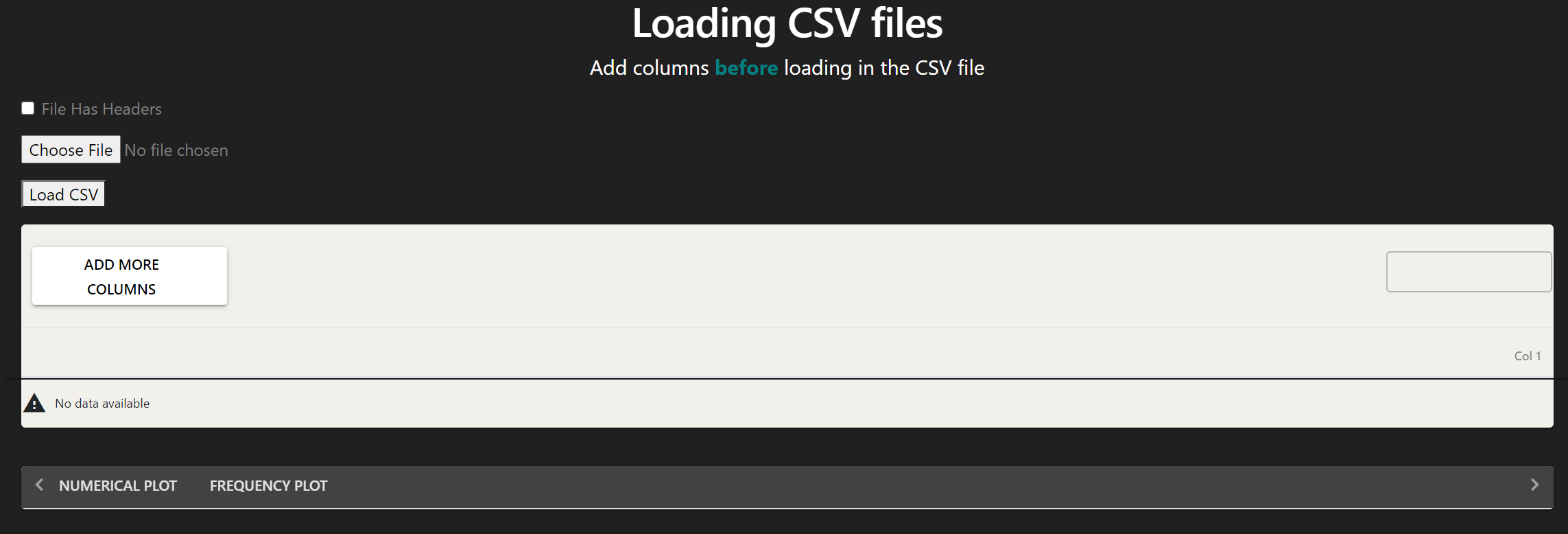
3. Scrollable List (top left)

* Filtered out the list of pageNames from facebook data and returned a list of unique pageNames. Sorted in alphabetical order.
* Created the search bar function that allows for user input. As user types in letters, the list would dynamically change to show the pageNames that contains the user input substring
* When user clicks on the pageName from the list, the selected pageName would case the whole dashboard to change. This includes the graph and the quasar table:

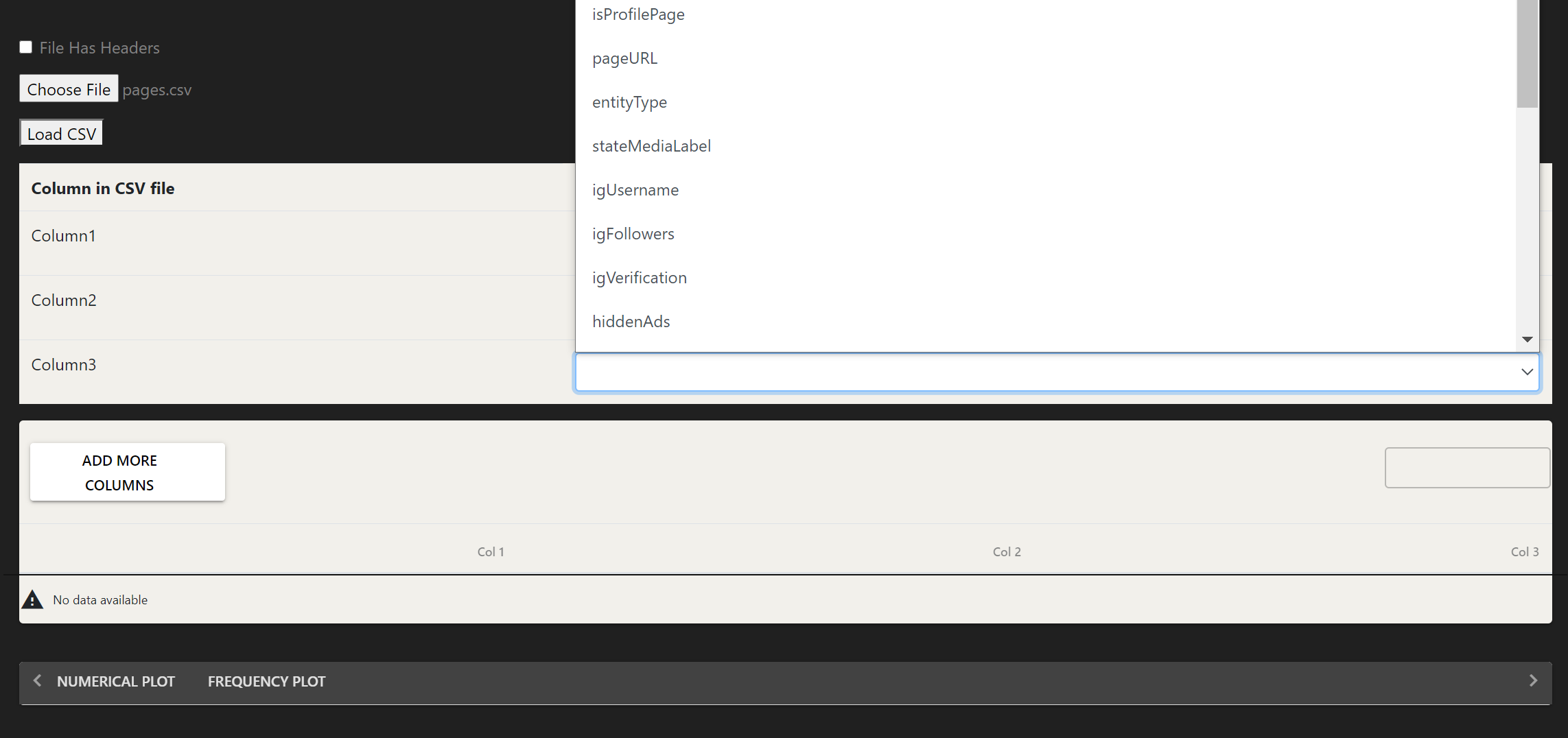


* For example, when the user types “sa” in the search box, and click on the “Sarangly” option, the graph shows the likes/ followers for the Sarangly pageName. The data in the table changes to show the Sarangly data only.
* The reset button in the quasar table allows for the default FB data to be reloaded.

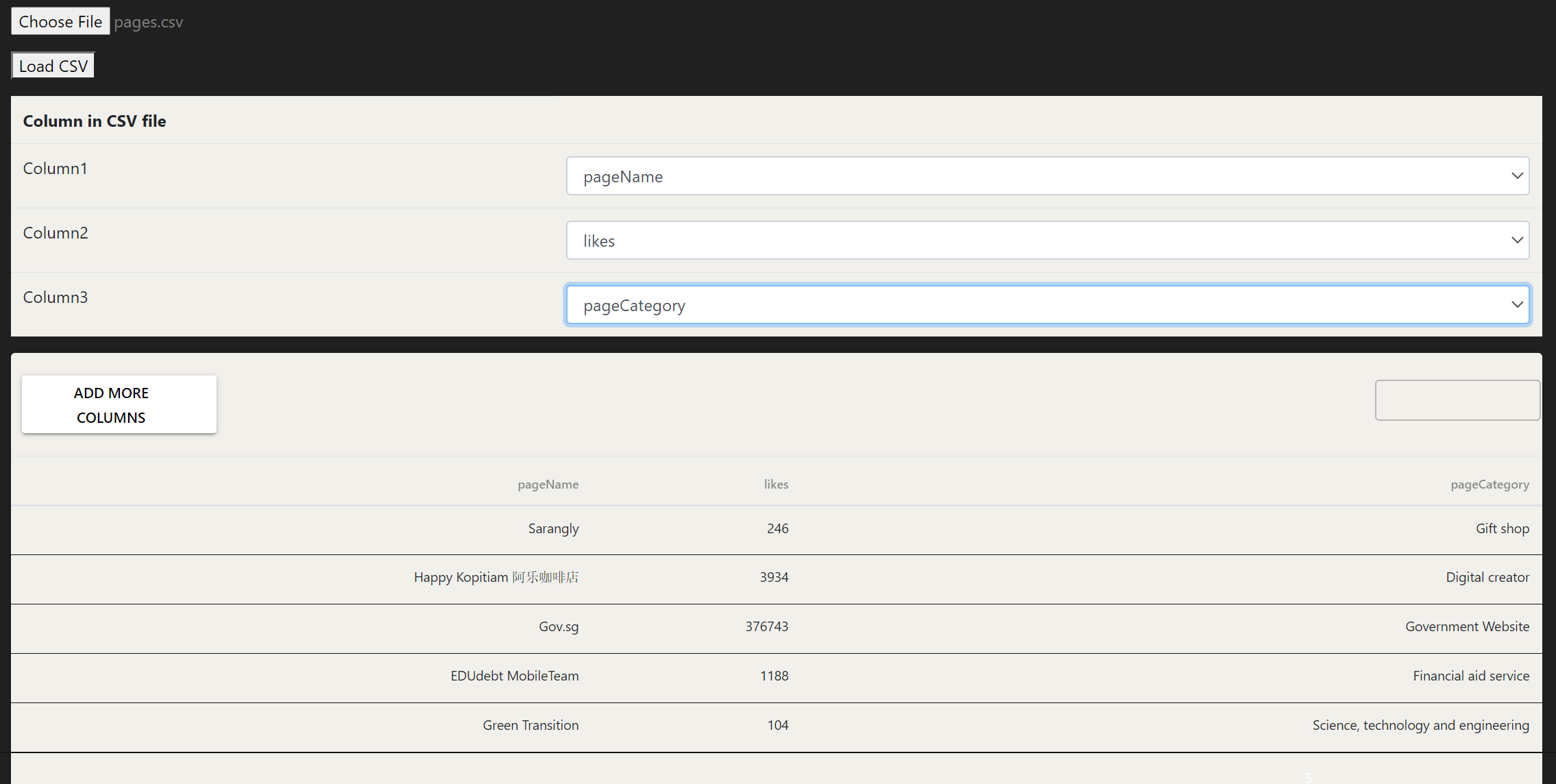
**LOADING CSV FILES**



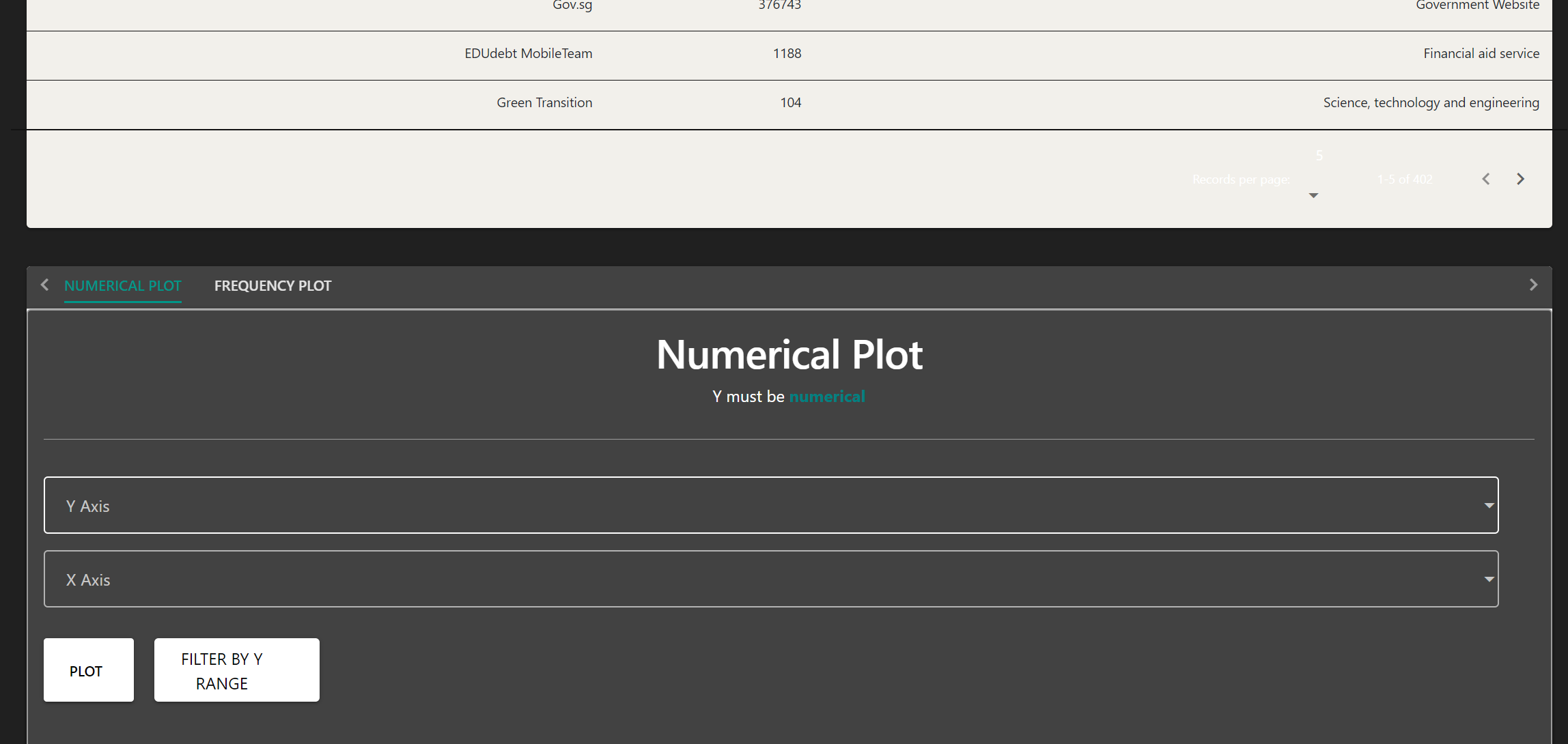
* Allows for users to directly load CSV files into the browser.
* They can choose the number of columns they want to look at using the “Add more columns” button
* After choosing their desired umber of columns and selecting their csv file using the “Choose file button”, they can choose exactly which columns of the CSV file they want to look at:



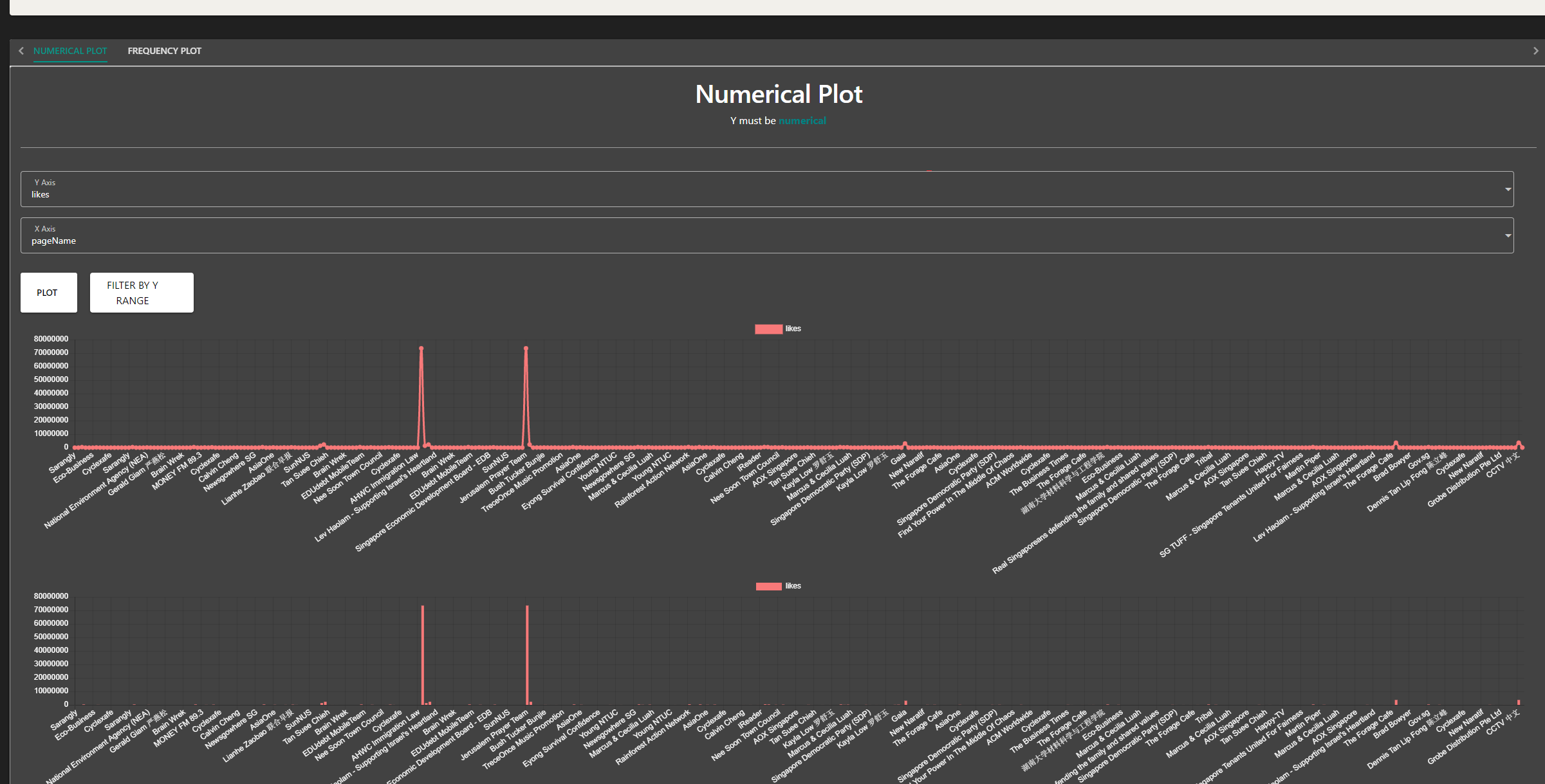
* After choosing all 3 columns for example, the data loads in the table below:



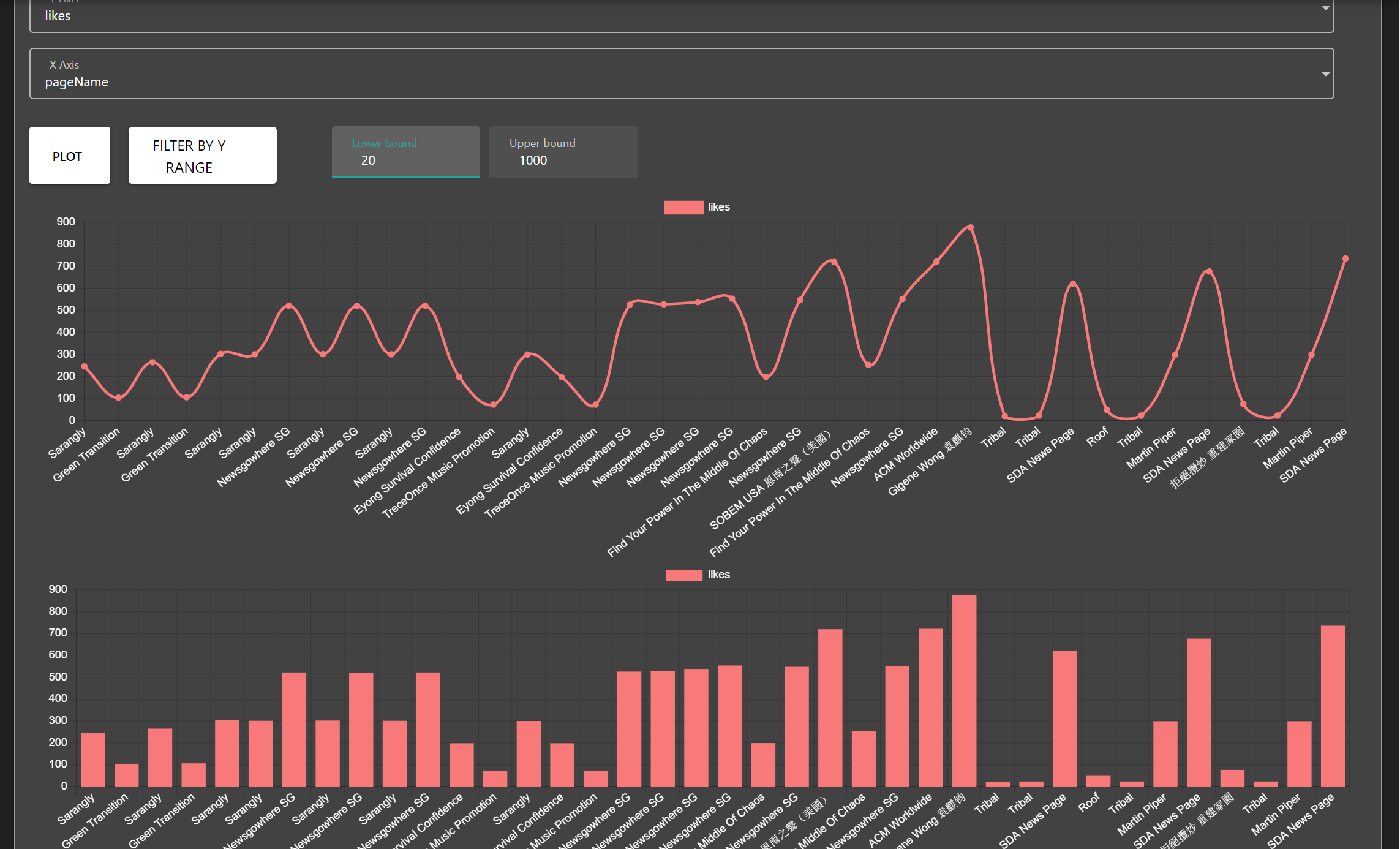
* The table headers also show the column name they chose instead of Col1, Col2 etc.
* Then more options appear below when the csv data is loaded:



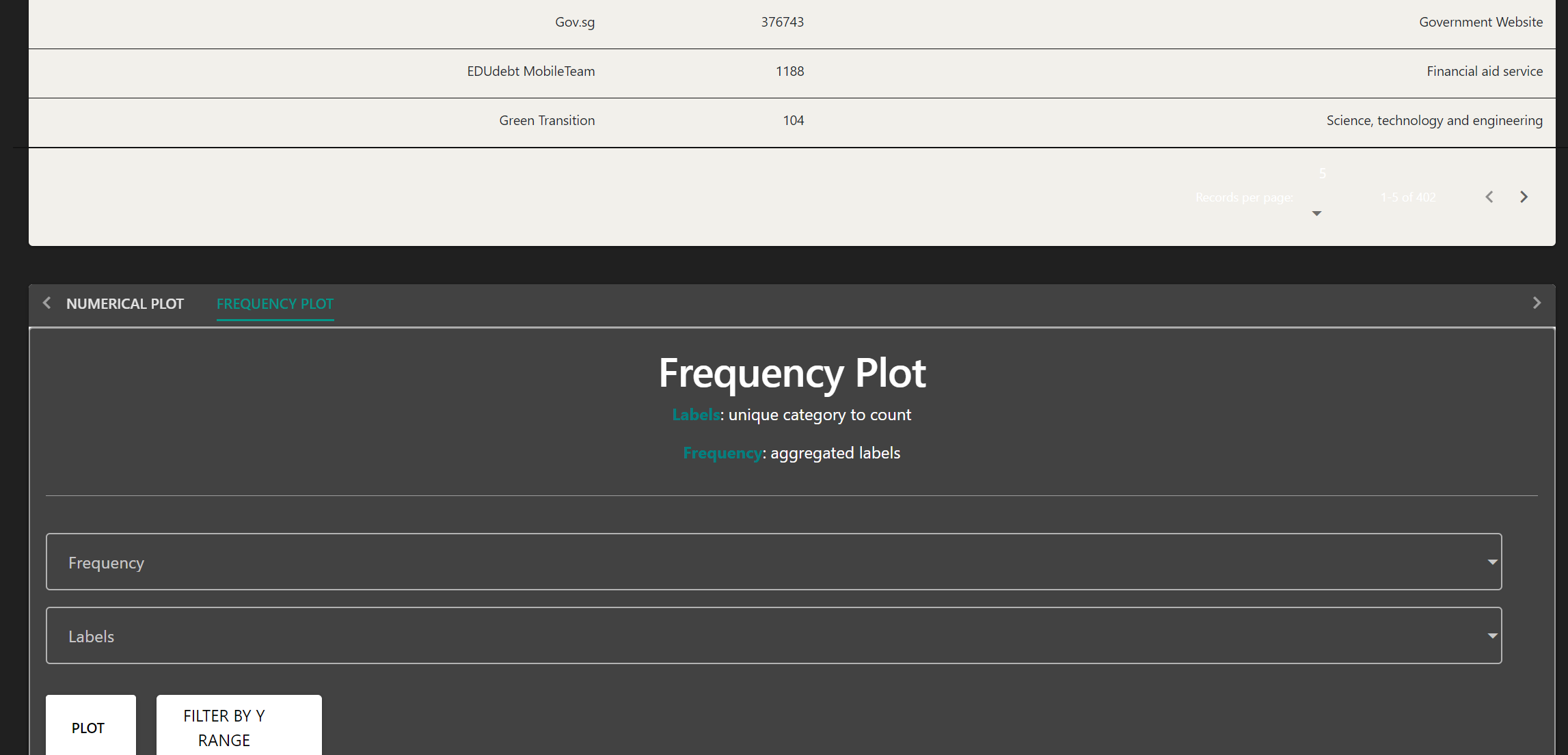
* It allows for a plot of any chosen column. However, the y data must be numerical. For example, I could plot “likes” against “pageNames” and press the “Plot” button. The following plot would be shown:



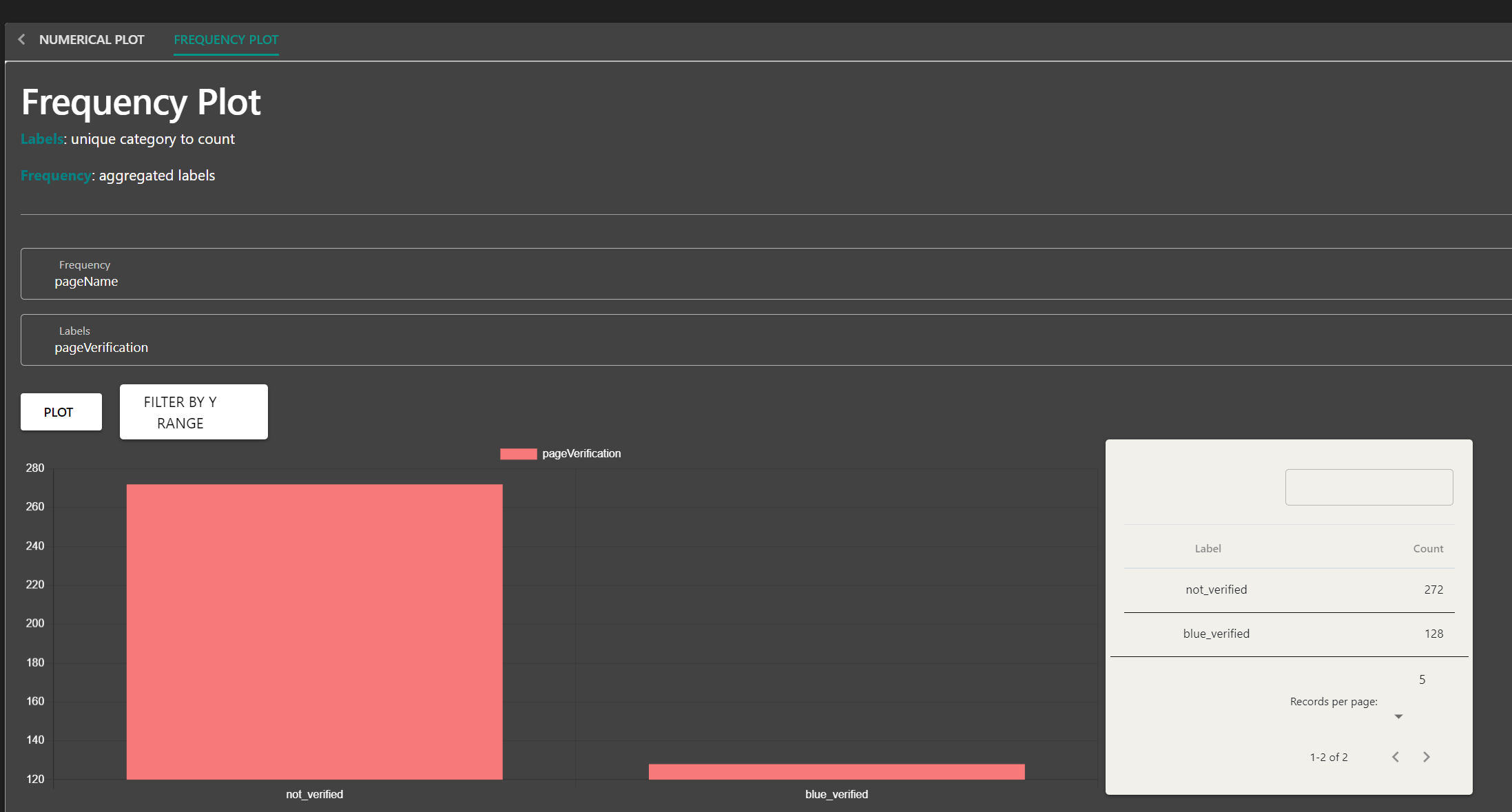
* It shows a Line and Bar chart. More charts can be added to the code if desired. However I thought a line and bar would sufficiently represent numerical data. Once again, the smaller valued data points are clustered together. Thus, I added the Filter by Y range function as well.
* Changing the input in the lower and upper bounds cause all graphs to change:



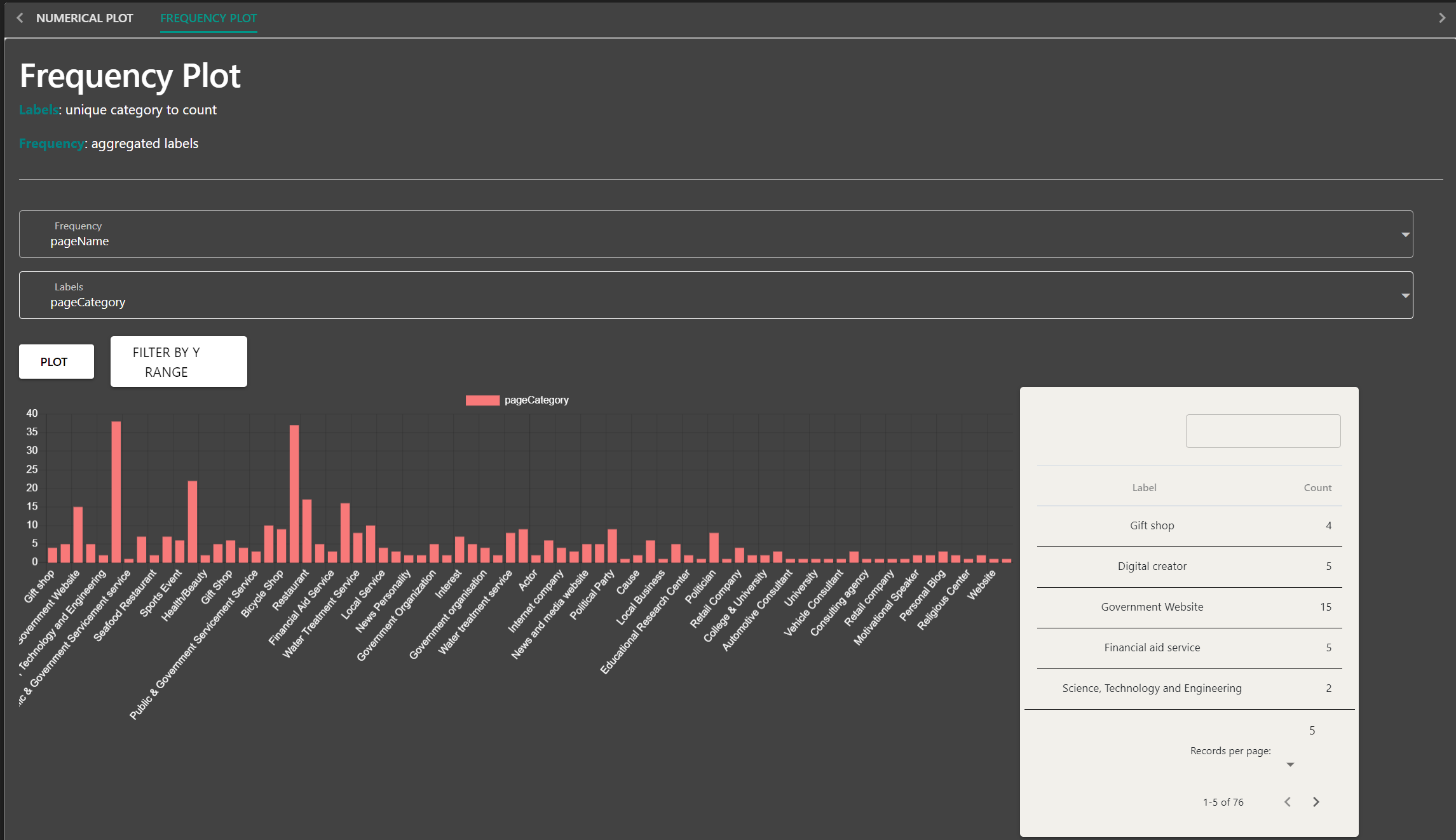
* Other than a Numerical Plot, a frequency plot can be done as well. I created tabs so that users can switch between which plots they want to visualise:



* Frequency plots take in the different labels for the different rows and creates an array of the distinct labels. Then it counts the number of rows that has that label. For example, for all pageNames, I want to count how many are True or False for pageVerification:



* This can be done for any other data that we want to count unique labels of. For example, how many types of pageCategory there are, counting frequency using pageNames:



* For expandability, more types of data visualisations can be added to the tabs section at the top.