CSCE 5320 Scientific Data Visualization
ICE-6
Marks and Channels

1. Created a covid cases dataset that contains the quantitative value and visualized it by creating a Bar Chart in VizHub.

Explanation-Introduction:

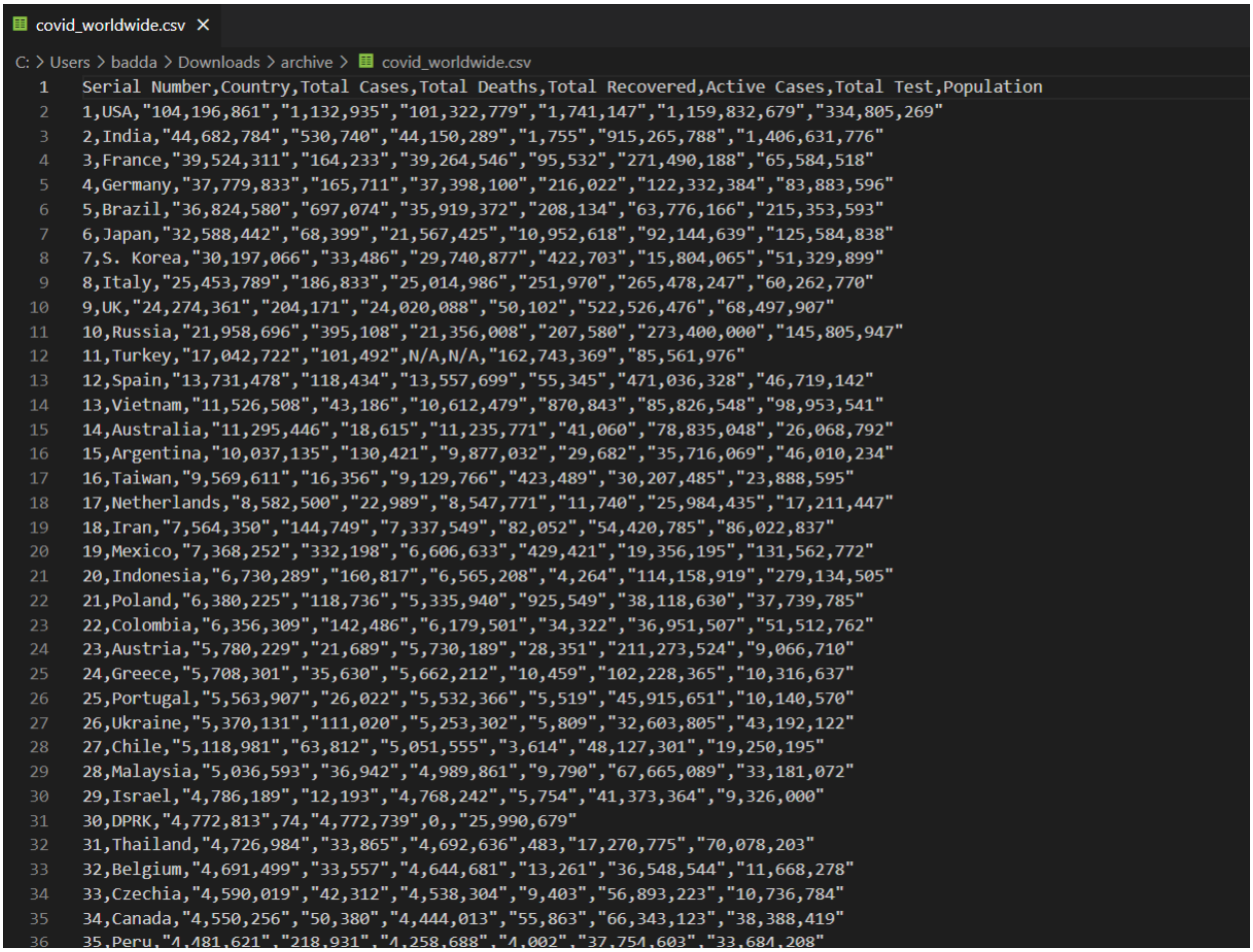
COVID-19 is an infectious illness that results from the SARS-CoV-2 virus. The provided dataset presents information about the deaths, recoveries, active cases, and total tests related to the coronavirus in different countries. The dataset was last updated one month ago and included eight columns: "Serial Number," "Country," "Total cases," "Total Deaths," "Total Recovered," "Active Cases," "Total Test," and "Population." There are approximately 232 rows in this dataset. We shall be using this dataset, cleaning it, and using the required attribute to visualize covid cases using a horizontal bar chart.

Methodology:

Firstly, I have downloaded Excel with Covid Cases data. We can see that it has eight columns: "Serial Number," "Country," "Total cases," "Total Deaths," "Total Recovered," "Active Cases," "Total Test," and "Population."

Serial Num	Country	Total Cases	Total Deat	Total Recovered	Active Cases	Total Test	Population
1	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269
2	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776
3	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518
4	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596
5	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593
6	Japan	32,588,442	68,399	21,567,425	10,952,618	92,144,639	125,584,838
7	S. Korea	30,197,066	33,486	29,740,877	422,703	15,804,065	51,329,899
8	Italy	25,453,789	186,833	25,014,986	251,970	265,478,247	60,262,770
9	UK	24,274,361	204,171	24,020,088	50,102	522,526,476	68,497,907
10	Russia	21,958,696	395,108	21,356,008	207,580	273,400,000	145,805,947
11	Turkey	17,042,722	101,492	N/A	N/A	162,743,369	85,561,976
12	Spain	13,731,478	118,434	13,557,699	55,345	471,036,328	46,719,142
13	Vietnam	11,526,508	43,186	10,612,479	870,843	85,826,548	98,953,541
14	Australia	11,295,446	18,615	11,235,771	41,060	78,835,048	26,068,792
15	Argentina	10,037,135	130,421	9,877,032	29,682	35,716,069	46,010,234
16	Taiwan	9,569,611	16,356	9,129,766	423,489	30,207,485	23,888,595
17	Netherlan	8,582,500	22,989	8,547,771	11,740	25,984,435	17,211,447
18	Iran	7,564,350	144,749	7,337,549	82,052	54,420,785	86,022,837
19	Mexico	7,368,252	332,198	6,606,633	429,421	19,356,195	131,562,772
20	Indonesia	6,730,289	160,817	6,565,208	4,264	114,158,919	279,134,505
21	Poland	6,380,225	118,736	5,335,940	925,549	38,118,630	37,739,785
22	Colombia	6,356,309	142,486	6,179,501	34,322	36,951,507	51,512,762
23	Austria	5,780,229	21,689	5,730,189	28,351	211,273,524	9,066,710
24	Greece	5,708,301	35,630	5,662,212	10,459	102,228,365	10,316,637
25	Portugal	5,563,907	26,022	5,532,366	5,519	45,915,651	10,140,570
26	Ukraine	5,370,131	111,020	5,253,302	5,809	32,603,805	43,192,122
27	Chile	5,118,981	63,812	5,051,555	3,614	48,127,301	19,250,195

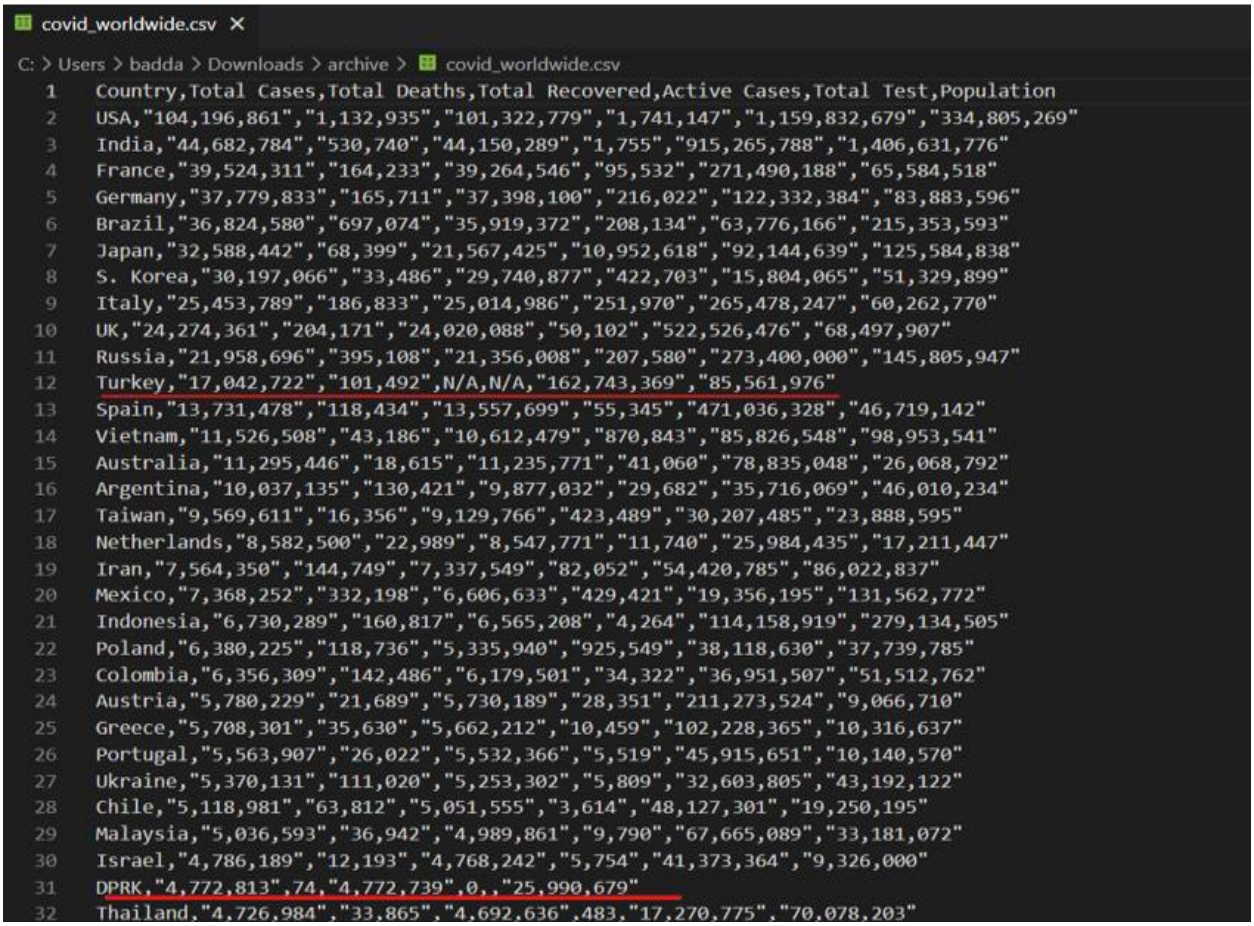
I have converted the excel into a CSV file as shown below.



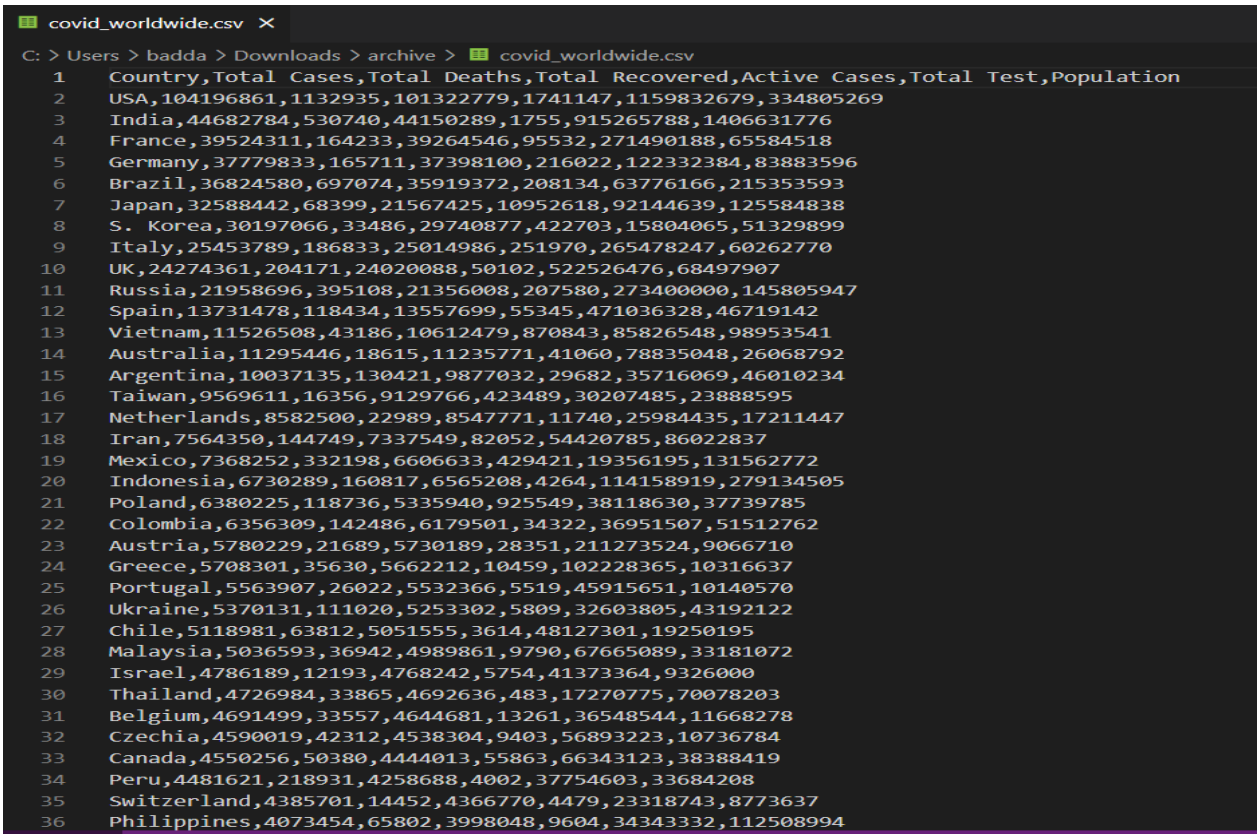
As we can see above, there is a serial number column that is not needed for the visualization or prediction of data. We shall remove that column. Below is what the dataset looks like after removing the column.

	A	B	C	D	E	F	G	H
1	Country	Total Cases	Total Deat	Total Recovered	Active Cases	Total Test	Population	
2	USA	104,196,861	1,132,935	101,322,779	1,741,147	1,159,832,679	334,805,269	
3	India	44,682,784	530,740	44,150,289	1,755	915,265,788	1,406,631,776	
4	France	39,524,311	164,233	39,264,546	95,532	271,490,188	65,584,518	
5	Germany	37,779,833	165,711	37,398,100	216,022	122,332,384	83,883,596	
6	Brazil	36,824,580	697,074	35,919,372	208,134	63,776,166	215,353,593	
7	Japan	32,588,442	68,399	21,567,425	10,952,618	92,144,639	125,584,838	
8	S. Korea	30,197,066	33,486	29,740,877	422,703	15,804,065	51,329,899	
9	Italy	25,453,789	186,833	25,014,986	251,970	265,478,247	60,262,770	
10	UK	24,274,361	204,171	24,020,088	50,102	522,526,476	68,497,907	
11	Russia	21,958,696	395,108	21,356,008	207,580	273,400,000	145,805,947	
12	Turkey	17,042,722	101,492	N/A	N/A	162,743,369	85,561,976	
13	Spain	13,731,478	118,434	13,557,699	55,345	471,036,328	46,719,142	
14	Vietnam	11,526,508	43,186	10,612,479	870,843	85,826,548	98,953,541	
15	Australia	11,295,446	18,615	11,235,771	41,060	78,835,048	26,068,792	
16	Argentina	10,037,135	130,421	9,877,032	29,682	35,716,069	46,010,234	
17	Taiwan	9,569,611	16,356	9,129,766	423,489	30,207,485	23,888,595	
18	Netherlan	8,582,500	22,989	8,547,771	11,740	25,984,435	17,211,447	
19	Iran	7,564,350	144,749	7,337,549	82,052	54,420,785	86,022,837	
20	Mexico	7,368,252	332,198	6,606,633	429,421	19,356,195	131,562,772	
21	Indonesia	6,730,289	160,817	6,565,208	4,264	114,158,919	279,134,505	
22	Poland	6,380,225	118,736	5,335,940	925,549	38,118,630	37,739,785	
23	Colombia	6,356,309	142,486	6,179,501	34,322	36,951,507	51,512,762	
24	Austria	5,780,229	21,689	5,730,189	28,351	211,273,524	9,066,710	
25	Greece	5,708,301	35,630	5,662,212	10,459	102,228,365	10,316,637	
26	Portugal	5,563,907	26,022	5,532,366	5,519	45,915,651	10,140,570	
27	Ukraine	5,370,131	111,020	5,253,302	5,809	32,603,805	43,192,122	
28	Chile	5,118,981	63,812	5,051,555	3,614	48,127,301	19,250,195	

We can observe that a few rows have many N/A and NULL values, removing such rows. Below are a few such rows that have been highlighted with a red underline.



I have converted the numbers in text format to numeric values, which will make it easy for visualization. Below is the final CSV file without any junk data in it. This clean data has 7 columns and 196 rows.

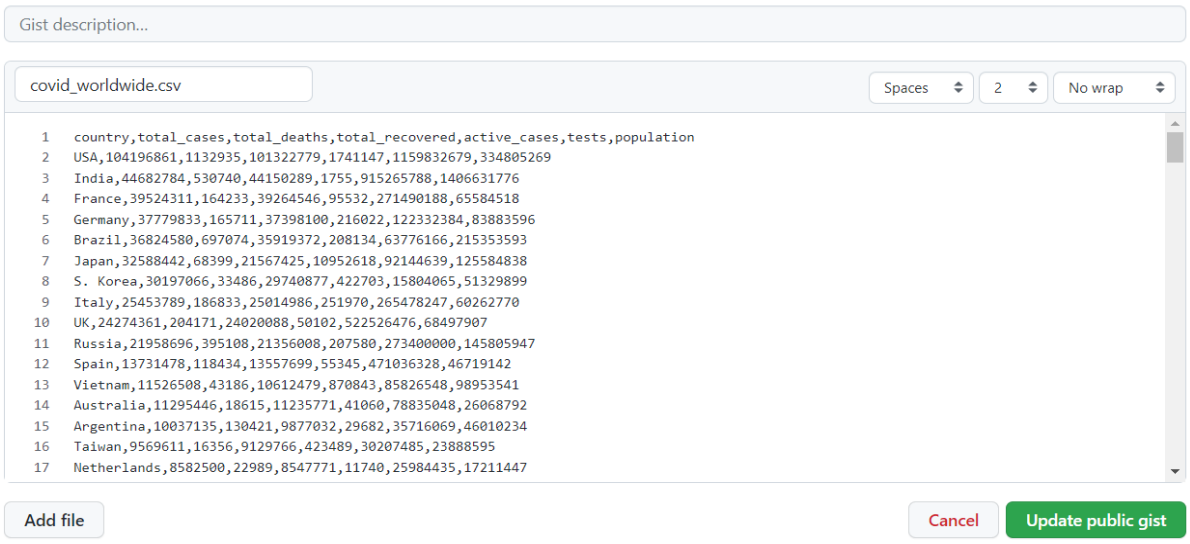


covid_worldwide.csv

C: > Users > badda > Downloads > archive > covid_worldwide.csv

```
Country,Total Cases,Total Deaths,Total Recovered,Active Cases,Total Test,Population
USA,104196861,1132935,101322779,1741147,1159832679,334805269
India,44682784,530740,44150289,1755,915265788,1406631776
France,39524311,164233,39264546,95532,271490188,65584518
Germany,37779833,165711,37398100,216022,122332384,83883596
Brazil,36824580,697074,35919372,208134,63776166,215353593
Japan,32588442,68399,21567425,10952618,92144639,125584838
S. Korea,30197066,33486,29740877,422703,15804065,51329899
Italy,25453789,186833,25014986,251970,265478247,60262770
UK,24274361,204171,24020088,50102,522526476,68497907
Russia,21958696,395108,21356008,207580,273400000,145805947
Spain,13731478,118434,13557699,55345,471036328,46719142
Vietnam,11526508,43186,10612479,870843,85826548,98953541
Australia,11295446,18615,11235771,41060,78835048,26068792
Argentina,10037135,130421,9877032,29682,35716069,46010234
Taiwan,9569611,16356,9129766,423489,30207485,23888595
Netherlands,8582500,22989,8547771,11740,25984435,17211447
Iran,7564350,144749,7337549,82052,54420785,86022837
Mexico,7368252,332198,6606633,429421,19356195,131562772
Indonesia,6730289,160817,6565208,4264,114158919,279134505
Poland,6380225,118736,5335940,925549,38118630,37739785
Colombia,6356309,142486,6179501,34322,36951507,51512762
Austria,5780229,21689,5730189,28351,211273524,9066710
Greece,5708301,35630,5662212,10459,102228365,10316637
Portugal,5563907,26022,5532366,5519,45915651,10140570
Ukraine,5370131,111020,5253302,5809,32603805,43192122
Chile,5118981,63812,5051555,3614,48127301,19250195
Malaysia,5036593,36942,4989861,9790,67665089,33181072
Israel,4786189,12193,4768242,5754,41373364,9326000
Thailand,4726984,33865,4692636,483,17270775,70078203
Belgium,4691499,33557,4644681,13261,36548544,11668278
Czechia,4590019,42312,4538304,9403,56893223,10736784
Canada,4550256,50380,4444013,55863,66343123,38388419
Peru,4481621,218931,4258688,4002,37754603,33684208
Switzerland,4385701,14452,4366770,4479,23318743,8773637
Philippines,4073454,65802,3998048,9604,34343332,112508994
```

Now we load this clean data into the GitHub Gist, I have created a GitHub Gist as “covid_worldwide.csv” and paste the CSV data from my computer onto the GitHub Gist as shown below. I have saved it as a public Gist.



Gist description...

covid_worldwide.csv

Spaces 2 No wrap

```
country,total_cases,total_deaths,total_recovered,active_cases,tests,population
USA,104196861,1132935,101322779,1741147,1159832679,334805269
India,44682784,530740,44150289,1755,915265788,1406631776
France,39524311,164233,39264546,95532,271490188,65584518
Germany,37779833,165711,37398100,216022,122332384,83883596
Brazil,36824580,697074,35919372,208134,63776166,215353593
Japan,32588442,68399,21567425,10952618,92144639,125584838
S. Korea,30197066,33486,29740877,422703,15804065,51329899
Italy,25453789,186833,25014986,251970,265478247,60262770
UK,24274361,204171,24020088,50102,522526476,68497907
Russia,21958696,395108,21356008,207580,273400000,145805947
Spain,13731478,118434,13557699,55345,471036328,46719142
Vietnam,11526508,43186,10612479,870843,85826548,98953541
Australia,11295446,18615,11235771,41060,78835048,26068792
Argentina,10037135,130421,9877032,29682,35716069,46010234
Taiwan,9569611,16356,9129766,423489,30207485,23888595
Netherlands,8582500,22989,8547771,11740,25984435,17211447
```

Add file Cancel Update public gist

After saving, the file looks as shown below. In the final file, we have seven columns: “country”, "total_cases," "total_deaths," "total_recovered," "active_cases," "total_test," and "population." It has 196 rows.

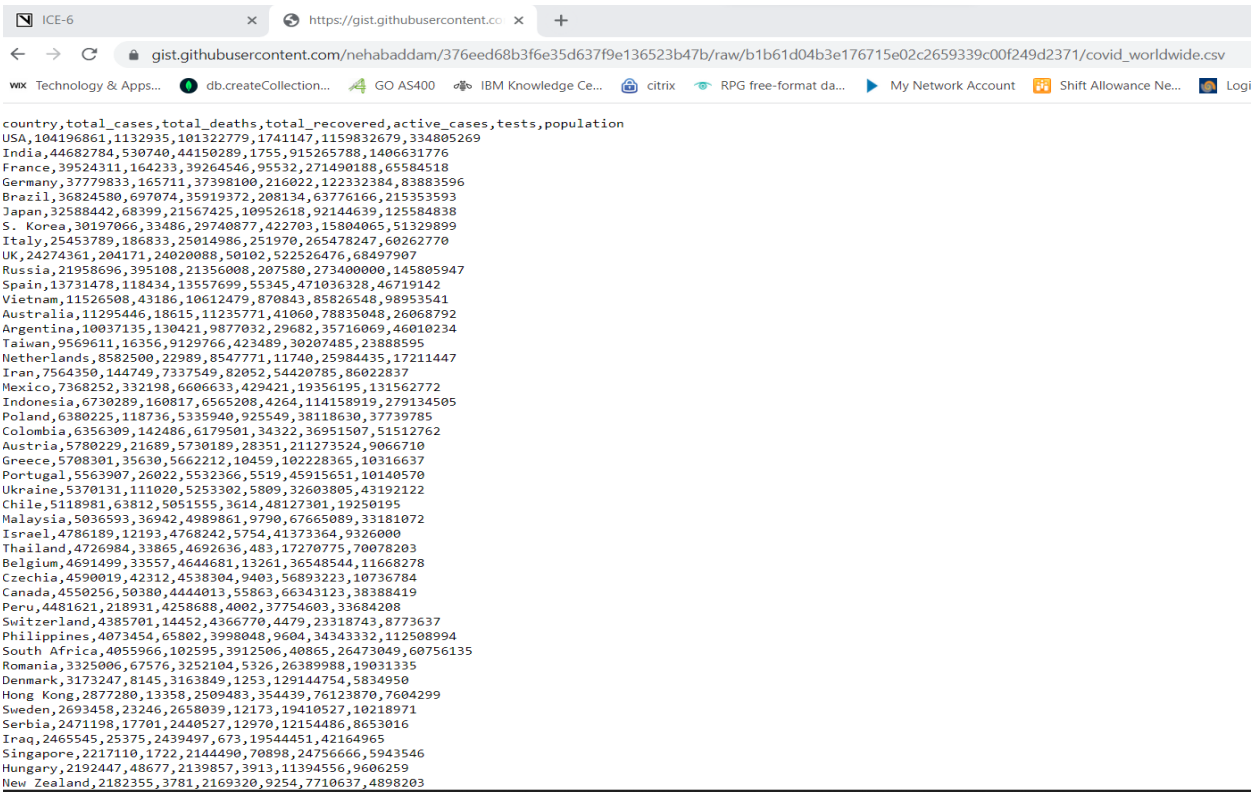
covid_worldwide.csv

Raw

Search this file...

	country	total_cases	total_deaths	total_recovered	active_cases	tests	population
2	USA	104196861	1132935	101322779	1741147	1159832679	334805269
3	India	44682784	530740	44150289	1755	915265788	1406631776
4	France	39524311	164233	39264546	95532	271490188	65584518
5	Germany	37779833	165711	37398100	216022	122332384	83883596
6	Brazil	36824580	697074	35919372	208134	63776166	215353593
7	Japan	32588442	68399	21567425	10952618	92144639	125584838
8	S. Korea	30197066	33486	29740877	422703	15804065	51329899
9	Italy	25453789	186833	25014986	251970	265478247	60262770
10	UK	24274361	204171	24020088	50102	522526476	68497907
11	Russia	21958696	395108	21356008	207580	273400000	145805947
12	Spain	13731478	118434	13557699	55345	471036328	46719142
13	Vietnam	11526508	43186	10612479	870843	85826548	98953541
14	Australia	11295446	18615	11235771	41060	78835048	26068792
15	Argentina	10037135	130421	9877032	29682	35716069	46010234
16	Taiwan	9569611	16356	9129766	423489	30207485	23888595
17	Netherlands	8582500	22989	8547771	11740	25984435	17211447
18	Iran	7564350	144749	7337549	82052	54420785	86022837
19	Mexico	7368252	332198	6606633	429421	19356195	131562772
20	Indonesia	6730289	160817	6565208	4264	114158919	279134505

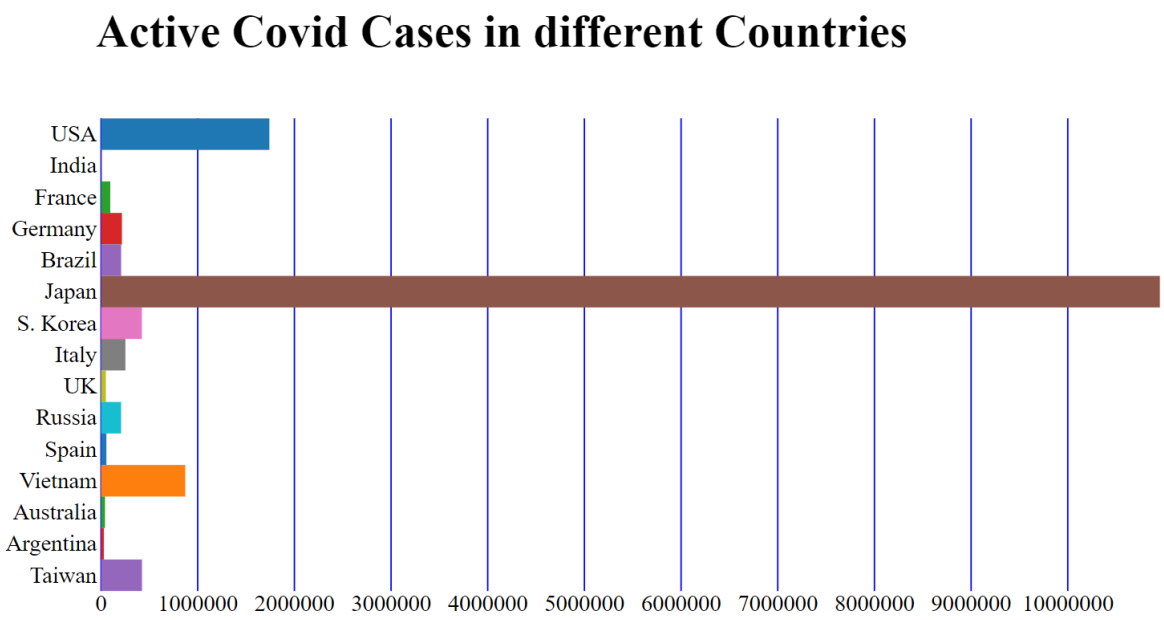
If we click the “Raw” button, the raw data is loaded on another tab of the browser as shown below.



I wrote a code using the D3 library in VizHub to create a horizontal bar chart that displays COVID-19 data for the top 15 countries with the most cases. Initially, I loaded the dataset using a gist URL, and only the active cases and countries were loaded. The height, width, and margins were all defined. To parse the data and convert the necessary columns to numbers, I passed a row function to the CSV function. The vertical scale, which maps the country names to their positions on the y-axis, was created using the scaleBand function. The horizontal scale, which maps the number of active cases to their positions on the x-axis, was created using the scaleLinear function. The message "Loading..." is displayed until the data is completely loaded. The returned function sends the visualized data as an SVG image. The xScale.ticks().the map function is used to create the tick marks and labels on the x-axis, and the yScale.domain().map function is used to create the labels on the y-axis. Finally, the data.map function is used to create rectangles for each country, representing active cases. The rectangles are positioned according to their country name on the y-axis and their respective data values on the x-axis using the translate function.

We use an HTML page to display the data. The code includes a style section with instructions for the body and message classes, defining font size, text alignment, and overflow. It also has a header with a first-level heading tag that presents the page's title. The body contains a div with an ID of "root" and a script tag linking to the bundle.js file. Furthermore, three script tags load

external libraries, React, ReactDOM, and D3. Below is how the bar chart is displayed. For example, Japan has the highest active cases of around 10000000. The next highest is the USA with around 2000000 cases.

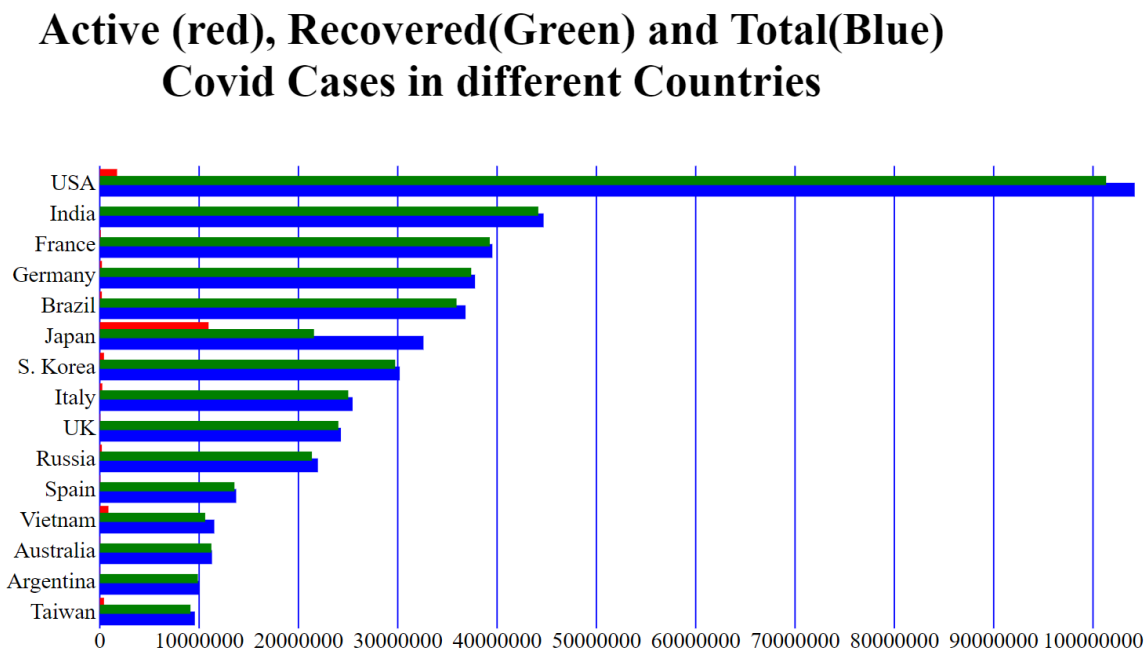


Now I have changed the code to include recovered cases and total cases. We are using the use effect function to get active, total, and recovered cases. I have changed the data.map function to create rectangles for each country, with different colors representing active cases(red), total cases(blue), and total recoveries(green). The setData function determines how many countries are displayed.

```
import React, { useState, useCallback, useEffect } from 'react';
import ReactDOM from 'react-dom';
import { select, axisLeft, axisBottom, csv, arc, pie, scaleBand, scaleLinear, scaleOrdinal, max, format, schemeCategory10 } from 'd3';


const csvUrl =
 'https://gist.githubusercontent.com/nehabaddam/376eed68b3f6e35d637f9e136523b47b/raw/b1b61d04b3e176715e02c2659339c00f249d2371/covid_worldwide.csv'
const width = 960;
const height = 500;
const margin = { top: 20, right: 20, bottom: 50, left: 200 };
const colorScale = scaleOrdinal(schemeCategory10);


const App = () => {
  const [data, setData] = useState(null);

  useEffect(() => {
    const row = d => {
      d.active_cases = +d['active_cases']
      d.total_cases = +d['total_cases']
      d.total_recovered = +d['total_recovered']
      return d;
    };
    csv(csvUrl, row).then(data => {
      setData(data.slice(0, 15));
    });
  }, []);

  if (!data) {
    return <pre>Loading...</pre>;
  }

  const innerHeight = height - margin.top - margin.bottom - 100;
  const innerWidth = width - margin.left - margin.right;
```

```
  const yScale = scaleBand()
    .domain(data.map(d => d.country))
    .range([0, innerHeight])
    .padding(0.1);

  const xScale = scaleLinear()
    .domain([0, max(data, d => d3.max([d.active_cases, d.total_cases, d.total_recovered]))])
    .range([0, innerWidth]);


return (
  <svg width={width} height={height}>
    <g transform={`translate(${margin.left},${margin.top})`}>
      {xScale.ticks().map(tickValue => (
        <g key={tickValue} transform={`translate(${xScale(tickValue)},0)`}>
          <line y2={innerHeight} stroke="blue" />
          <text style={{ textAnchor: 'middle' }} dy=".70em" y={innerHeight + 3}>
            {tickValue}
          </text>
        </g>
      ))}
      {yScale.domain().map(tickValue => (
        <text
          key={tickValue}
          style={{ textAnchor: 'end' }}
          x={-3}
          dy=".32em"
          y={yScale(tickValue) + yScale.bandwidth() / 2}
        >
          {tickValue}
        </text>
      ))}
      {data.map(d => (
        <g key={d.country} transform={`translate(0,${yScale(d.country)})`}>
          <rect
            x={0}
            y={0}
            width={xScale(d.active_cases)}
            height={yScale.bandwidth() / 2}
            fill="red"
          />
          <rect
            x={0}
            y={yScale.bandwidth() / 2}
            width={xScale(d.total_cases)}
            height={yScale.bandwidth() / 2}
            fill="blue"
          />

          <rect
            x={0}
            y={yScale.bandwidth() / 4}
            width={xScale(d.total_recovered)}
            height={yScale.bandwidth() / 3}
            fill="green"
          />

        </g>
      ))}
    </g>
  </svg>
);

};
const rootElement = document.getElementById('root');
ReactDOM.render(<App />, rootElement);
```

We can even visualize multiple columns in the dataset against the countries. Here in the below bar chart, we can observe total cases, recovered cases, and active cases. The number of cases is represented on the y-axis and countries are represented on the y-axis. We can observe that the USA has the highest total and recovered cases of around 100000000. Japan has the highest active cases of around 10000000. The active cases are represented using red columns, recovered ones are represented using green columns and total cases are represented using blue columns.



Conclusion:

To summarize, the dataset given to us contains COVID-19 data for various countries, including deaths, recoveries, active cases, and total tests. We cleaned the dataset by removing unnecessary columns, eliminating rows with null values, and converting text into numeric values. The cleaned dataset was then loaded into a GitHub Gist, and we utilized the D3 library to create a horizontal bar chart that displays COVID-19 data for the top 15 countries with the highest number of cases. We used HTML to display the chart with style instructions, including a header with a first-level heading tag that presents the page's title. Additionally, we modified the code to display active, recovered, and total cases for each country, and we used the useEffect function to retrieve the necessary data. Finally, we created a bar chart that represents active cases in red, recovered cases in green, and total cases in blue, enabling us to monitor and mitigate the spread of COVID-19 effectively.

Link to data:

https://gist.githubusercontent.com/nehabaddam/376eed68b3f6e35d637f9e136523b47b/raw/b1b61d04b3e176715e02c2659339c00f249d2371/covid_worldwide.csv

VizHub Link:
https://vizhub.com/nehabaddam/5d189bd18f384ad9b9db776ad7a76de0?edit=files&file=index.js

VizHub ID: nehabaddam

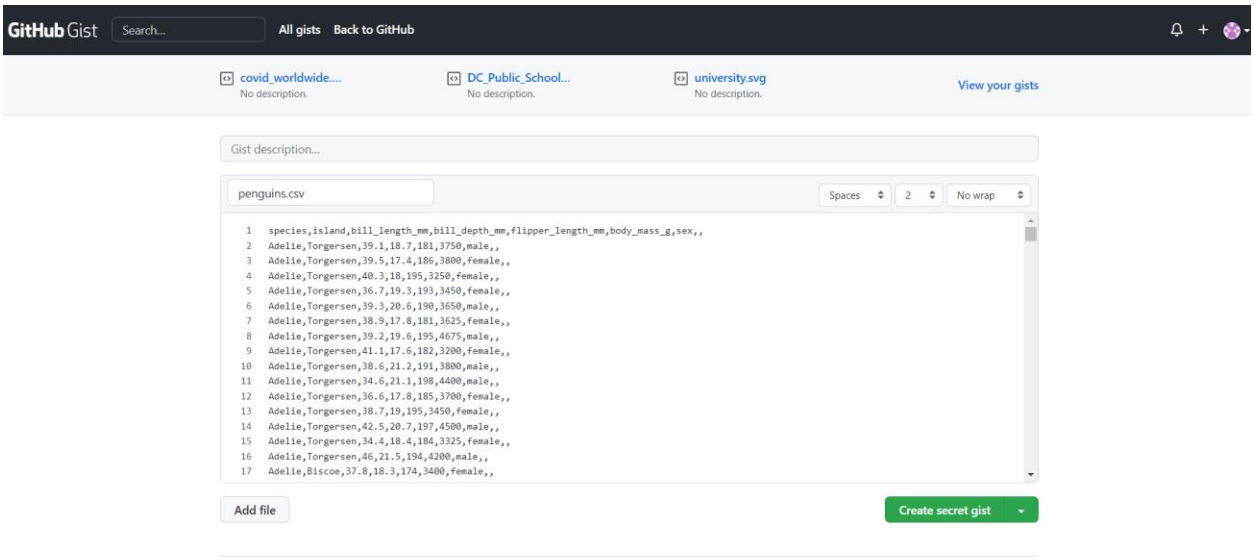
2. I have downloaded the penguin data set and created a scatter plot based on the data in VizHub.

Explanation-Introduction:

We are using penguin data set to create a scatter plot. This dataset consists of physical measurements of 3 different species of penguins on different islands. The measurements include bill length, bill depth, flipper length, body mass, and sex of the penguins. The bill length is measured in millimeters, while body mass is measured in grams. The dataset contains information on both male and female penguins. The scatter plot represents flipper length against bill length with body mass for different species represented using different colors.

Methodology:

I have downloaded the penguin dataset and added it to my gist. I have used this data to create a scatter plot with X and Y labels for the plot.



The dataset has 8 columns, “species”, “island”, “bill_length_mm”, “flipper_length_mm”, “body_mass_g” and “sex”. There are 333 rows of data.

penguins.csv

Raw

Search this file...

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
2	Adelie	Torgersen	39.1	18.7	181	3750	male
3	Adelie	Torgersen	39.5	17.4	186	3800	female
4	Adelie	Torgersen	40.3	18	195	3250	female
5	Adelie	Torgersen	36.7	19.3	193	3450	female
6	Adelie	Torgersen	39.3	20.6	190	3650	male
7	Adelie	Torgersen	38.9	17.8	181	3625	female
8	Adelie	Torgersen	39.2	19.6	195	4675	male
9	Adelie	Torgersen	41.1	17.6	182	3200	female
10	Adelie	Torgersen	38.6	21.2	191	3800	male
11	Adelie	Torgersen	34.6	21.1	198	4400	male
12	Adelie	Torgersen	36.6	17.8	185	3700	female
13	Adelie	Torgersen	38.7	19	195	3450	female
14	Adelie	Torgersen	42.5	20.7	197	4500	male
15	Adelie	Torgersen	34.4	18.4	184	3325	female
16	Adelie	Torgersen	46	21.5	194	4200	male
17	Adelie	Biscoe	37.8	18.3	174	3400	female
18	Adelie	Biscoe	37.7	18.7	180	3600	male
19	Adelie	Biscoe	35.9	19.2	189	3800	female
20	Adelie	Biscoe	38.2	18.1	185	3950	male

Below is the code, it uses D3.js, a widely-used JavaScript library for data visualization, to generate a scatter plot of penguin data. The data is contained in a CSV file and is loaded into the script with D3's csv function. A parseRow function is utilized to convert specific columns to numerical values.

The x and y scales for the plot are set using D3's scaleLinear function, with the flipper length and body mass columns used for the domain, respectively. The characteristics for each data point, including the x and y coordinates, species, color, bill length, and depth values, are established using an array of objects.

The plot itself is generated by utilizing D3's select, append, and join functions to create and format circles for each data point. The axis labels are also generated using D3's append function and repositioned with the translate and rotate functions.

Overall, this code presents a fundamental template for creating scatter plots with D3.js and can be customized for different datasets by modifying the CSV file and adapting the parseRow, xValue, yValue, and marker functions to match the corresponding data columns.

```
<!DOCTYPE html>
<html>
  <head>
    <title>ICE-6 Penguin Datset</title>
    <style>
      body {
        margin: 0;
        overflow: hidden;
      }
      .tick text {
        font-size: 23px;
      }
    </style>
    <script src="https://unpkg.com/d3@6.7.0/dist/d3.min.js"></script>
  </head>
  <body>
    <script>
const {csv, select, scaleLinear, extent, axisLeft, axisBottom, } = d3;
const csvUrl = 'https://gist.githubusercontent.com/nehabaddam/b43f07ea239cff419dac8237e26bdd69/raw/018bb6fd3afa5a65afb9c2ccbeb5b1c5b48238fb/penguins.csv'
      const parseRow = (d) => {
        d.  bill_length_mm = +d.  bill_length_mm;
        d.  bill_depth_mm = +d. bill_depth_mm;
        d.flipper_length_mm = +d.flipper_length_mm;
        d.body_mass_g = +d.body_mass_g;
        return d;
      };  //LOAD THE DATA
      const xValue = (d) => d.flipper_length_mm;
      const yValue = (d) => d.body_mass_g;


      const rxValue = (d) => d. bill_length_mm;
      const ryValue = (d) => d. bill_depth_mm;

      const speciesValue = (d) => d.species;
```

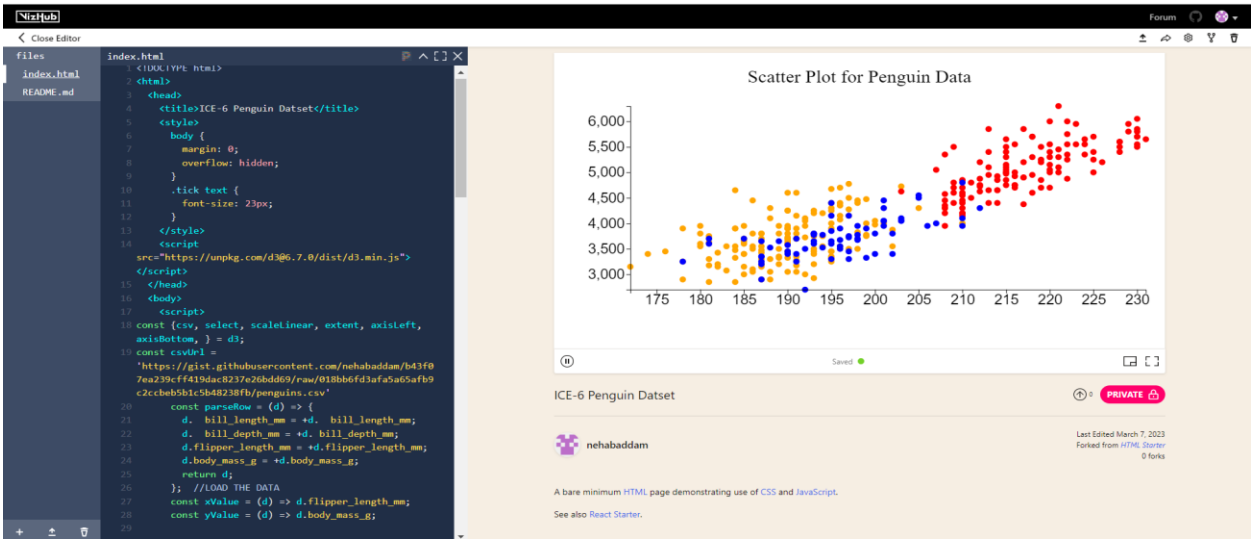
```
      const margin = {
        top: 90,
        right: 30,
        bottom: 100,
        left: 120,
      };
      const radius = 5;

      const width = window.innerWidth;
      const height = window.innerHeight;
      const svg = select('body')
        .append('svg')
        .attr('width', width)
        .attr('height', height);


      const main = async () => {
        const data = await csv(csvUrl, parseRow);

        const x = scaleLinear()
          .domain(extent(data, xValue))
          .range([margin.left, width - margin.right]);

        const y = scaleLinear()
          .domain(extent(data, yValue))
          .range([height - margin.bottom, margin.top]);
```

```
const marker = (d) => {
  switch(d){
    case 'Adelie':
      return 'orange';
    case 'Gentoo':
      return 'red';
    case 'Chinstrap':
      return 'blue';
    default:
      return 'green';
  }
}

const marks = data.map((d) => ({
  x: x(xValue(d)),
  y: y(yValue(d)),
  species: speciesValue(d),
  color: marker(speciesValue(d)),
  rx: rxValue(d),
  ry: ryValue(d),
}));

svg
  .selectAll('circle')
  .data(marks)
  .join('circle')
  .attr('cx', (d) => d.x)
  .attr('cy', (d) => d.y)
  .attr('fill', (d) => d.color)
  .attr('rx', (d) => (d.rx))
  .attr('ry', (d) => (d.ry))
  .attr('r', radius);
```

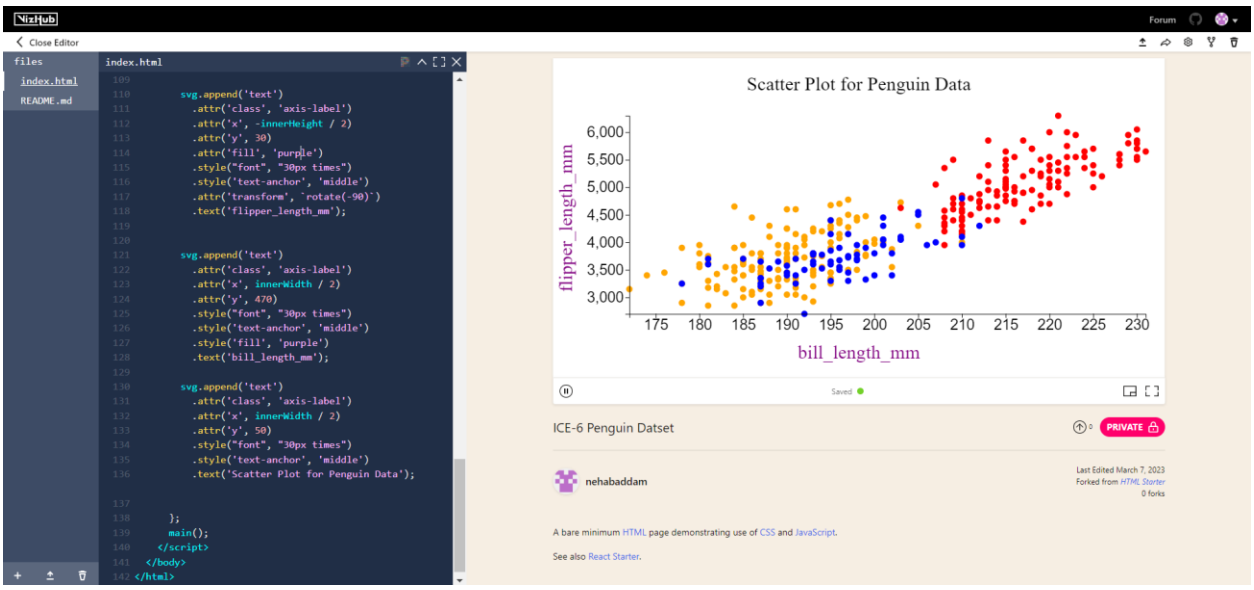
```
svg
  .append('g')
  .attr('transform', `translate(${margin.left},0)`)
  .call(axisLeft(y));


svg
  .append('g')
  .attr(
    'transform',
    `translate(0,${height - margin.bottom})`
  )
  .call(axisBottom(x));

svg.append('text')
  .attr('class', 'axis-label')
  .attr('x', -innerHeight / 2)
  .attr('y', 30)
  .attr('fill', 'purple')
  .style("font", "30px times")
  .style('text-anchor', 'middle')
  .attr('transform', `rotate(-90)`)
  .text('flipper_length_mm');


svg.append('text')
  .attr('class', 'axis-label')
  .attr('x', innerWidth / 2)
  .attr('y', 470)
  .style("font", "30px times")
  .style('text-anchor', 'middle')
  .style('fill', 'purple')
  .text('bill_length_mm');
```

```
        svg.append('text')
          .attr('class', 'axis-label')
          .attr('x', innerWidth / 2)
          .attr('y', 470)
          .style("font", "30px times")
          .style('text-anchor', 'middle')
          .style('fill', 'purple')
          .text('bill_length_mm');

        svg.append('text')
          .attr('class', 'axis-label')
          .attr('x', innerWidth / 2)
          .attr('y', 50)
          .style("font", "30px times")
          .style('text-anchor', 'middle')
          .text('Scatter Plot for Penguin Data');

      };
      main();
    </script>
  </body>
</html>
```

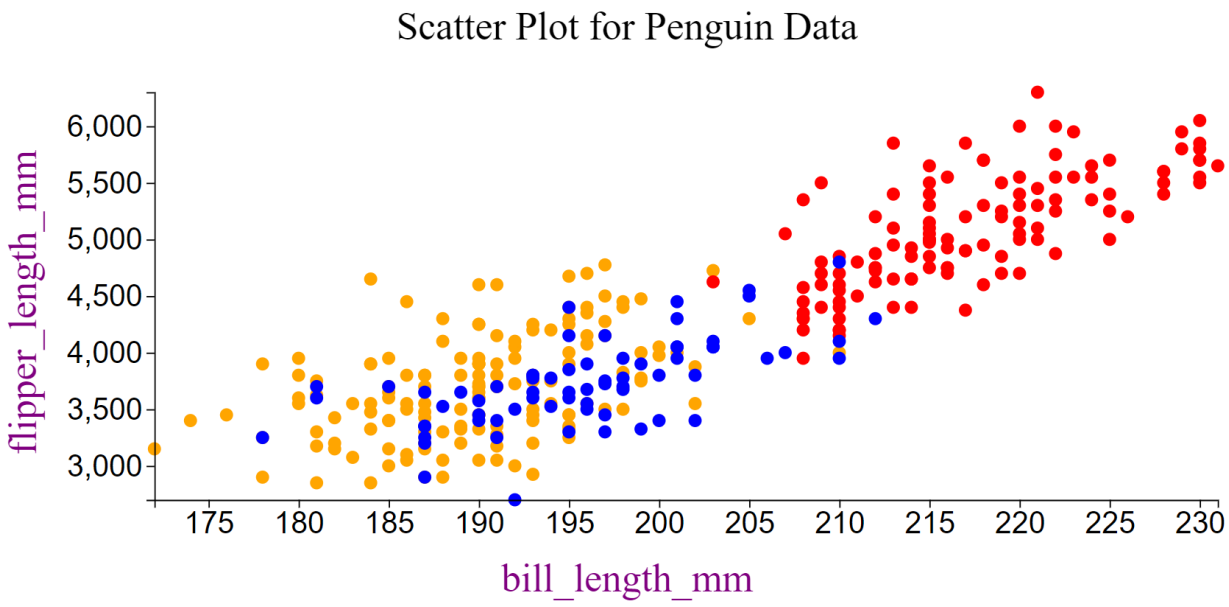
Below is how the to scatter plot appears without labels initially.



After adding the labels, the scatter plot is displayed as shown below.



The x-axis represents flipper_length_mm and the y-axis represents bill_length_mm. The 3 species of penguins are represented using different colors, the Adelie species uses orange, Gentoo is red, and Chinstrap is blue.



Conclusion:

In conclusion, the code creates a scatter plot using the D3.js library. The scatter plot visualizes data from a penguin dataset hosted on GitHub, where each point represents a penguin's body mass and flipper length. The color of each point corresponds to the penguin's species. The code also includes axes with labels for both the x and y-axis, as well as a title for the plot. The plot is designed to be responsive, adapting to the size of the window.

Link to Penguin Dataset:

https://gist.githubusercontent.com/nehabaddam/b43f07ea239cff419dac8237e26bdd69/raw/018bb6fd3afa5a65afb9c2ccbeb5b1c5b48238fb/penguins.csv

VizHub Link:

https://vizhub.com/nehabaddam/8cc9cc5d75f440928bc1e65e46355cd0?edit=files&file=index.html

VizHub ID: nehabaddam