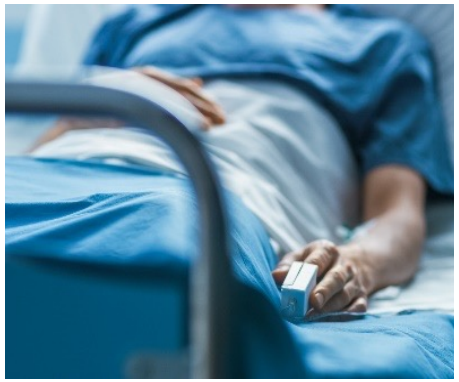
Coronavirus Pandemic (COVID-19) Dashboard

PROJECT 3

HANNAH HILL – SAHAR JAMAL – NEDA MEHDIZADEH



The combination of getting vaccinated and following CDC recommendations to protect yourself and others offers the best protection from COVID-19.

- Cover your nose and mouth with a mask.
- Stay at least 6 feet from people who don't live with you.
- Avoid crowds and poorly ventilated indoor spaces.
- Wash your hands.

Getting vaccinated can help prevent getting sick with COVID-19

Getting vaccinated can help prevent getting sick with COVID-19
COVID-19 vaccines will not cause you to test positive on COVID-19 viral tests

COVID-19 and Vaccine Data Dashboard

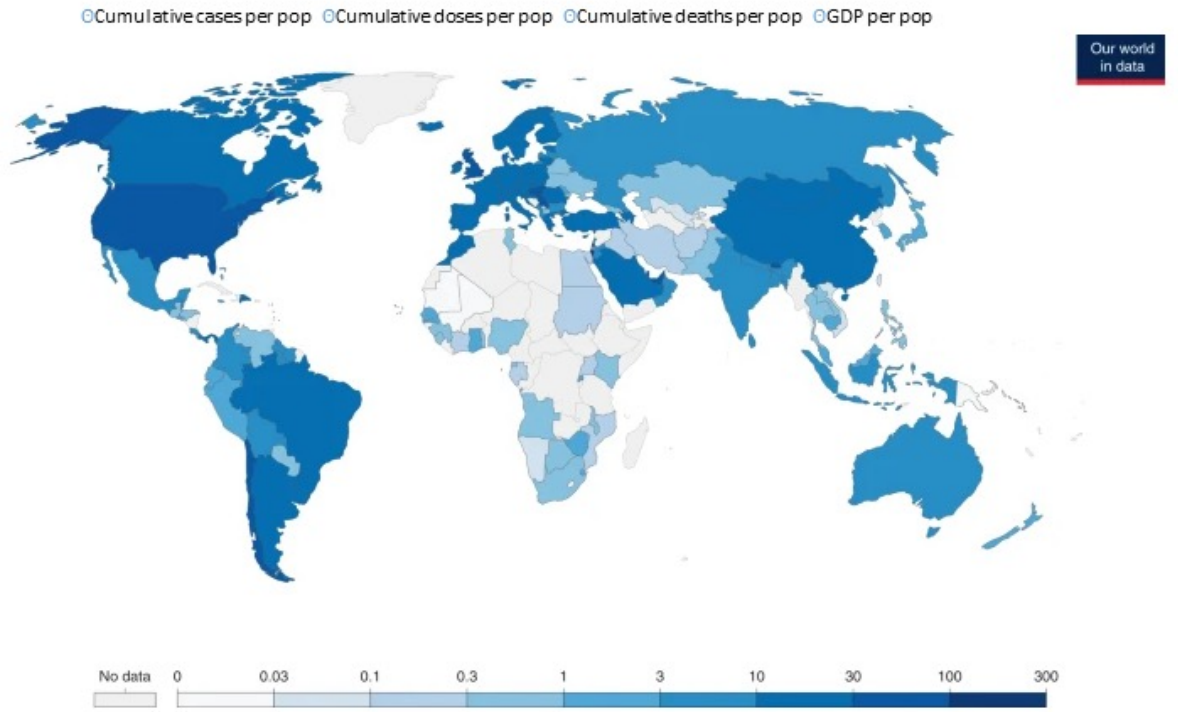
In this dashboard we present trends for the 2020-2021 data regarding COVID-19:

- Get an overview of the pandemic for any country on a single page
- Explore the global datasets on covid vaccination
- Explore the data confirmed Covid-19 cases for all countries
- Explore metrics like total life expectancy, total hospital beds and human development index in each country

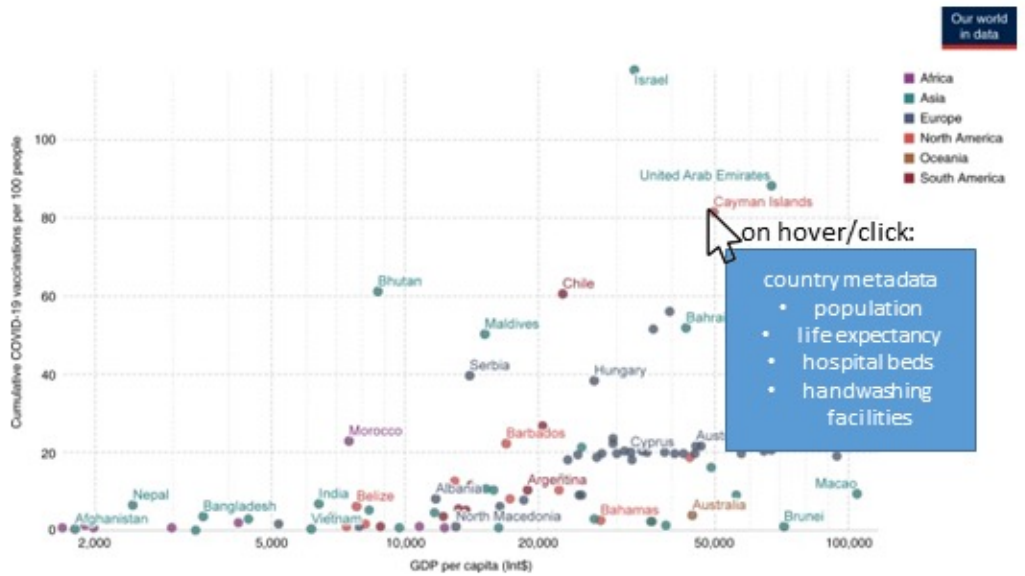
Our dashboard would be useful for travelers or those interested in researching the COVID-19 pandemic and its trends.

Dashboard Vision

Worldwide COVID-19 Data Dashboard



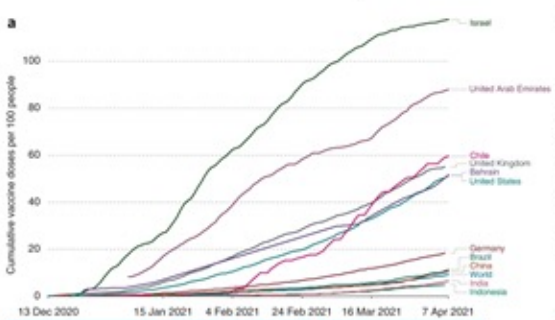
y-axis options:
- cumulative cases
- cumulative doses
- cumulative deaths
- excess mortality



on hover/click:

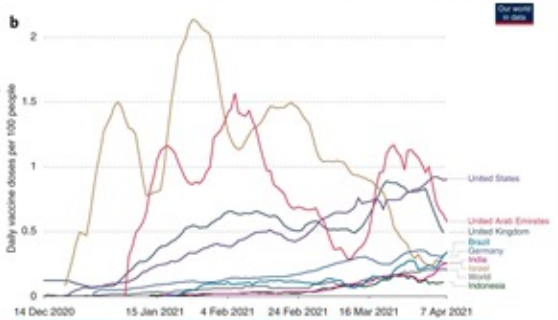
country metadata
- population
- life expectancy
- hospital beds
- handwashing facilities

cumulative cases/doses



x-axis slide

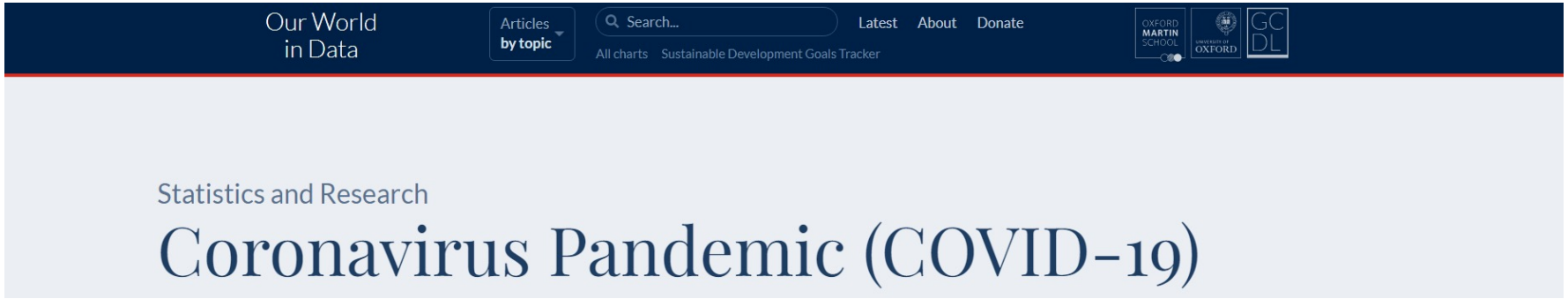
daily cases/doses/stringency



x-axis slide

country checkboxes

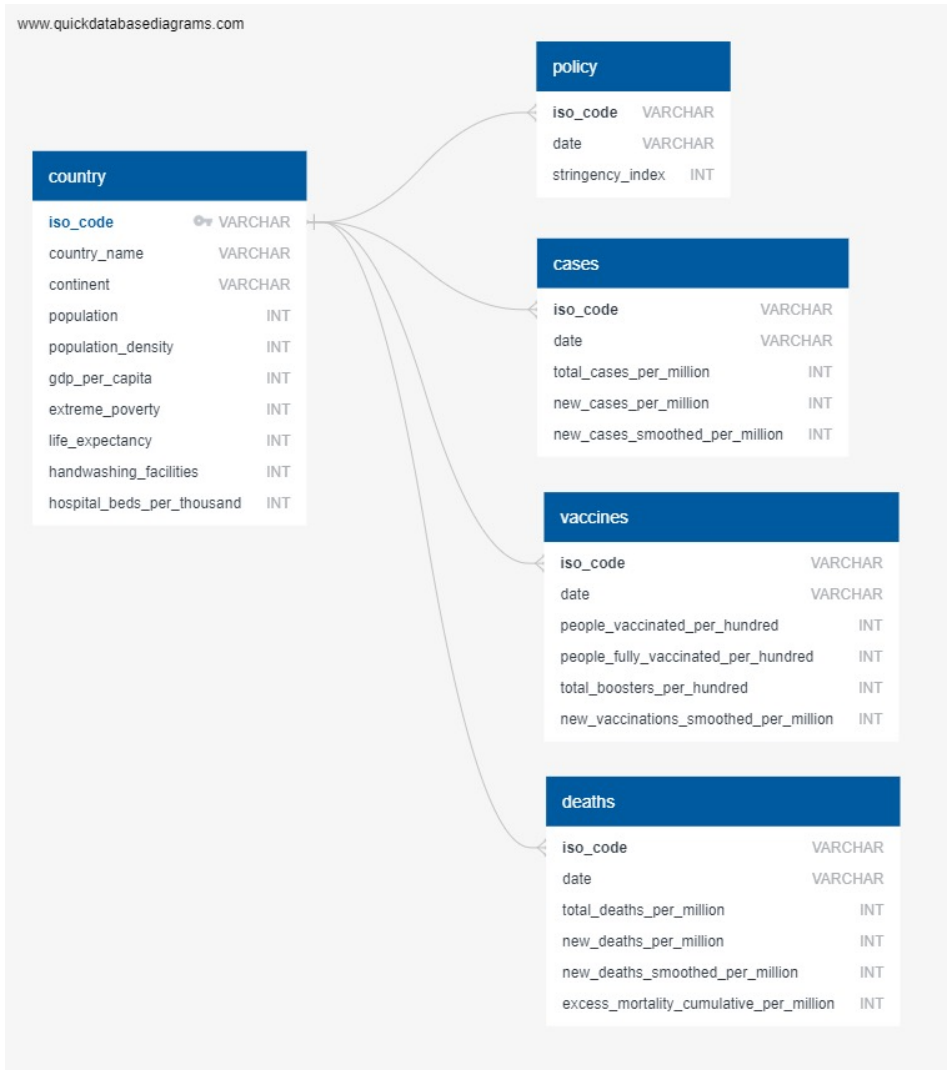
Data Source



We obtained Data as a csv file from : https://ourworldindata.org/coronavirus

and https://github.com/owid/covid-19-data/tree/master/public/data

Design and Build Database




```
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, func
engine = create_engine(f'postgresql://postgres:{password}@localhost:5432/covid19_db')
conn = engine.connect()
```

```
base = automap_base()
```

```
base.prepare(engine, reflect=True)
```

```
base.classes.keys()
```

```
['country']
```

```
country.to_sql(name='country', con=engine, if_exists='append', index=False)
```

```
49  SELECT * FROM country;
```

Data Output | Explain | Messages | Notifications

	iso_code [PK] character varying	country_name character varying	continent character varying	population bigint	population_density integer	gdp_per_capita integer	extreme_poverty integer	life_expectancy integer	handwashing_facilities integer	hospital_beds_per_thousand integer
1	AFG	Afghanistan	Asia	39835428	54	1804	[null]	65	38	1
2	OWID_AFR	Africa	[null]	1373486472	[null]	[null]	[null]	[null]	[null]	[null]
3	ALB	Albania	Europe	2872934	105	11803	1	79	[null]	3
4	DZA	Algeria	Africa	44616626	17	13914	1	77	84	2
5	AND	Andorra	Europe	77354	164	[null]	[null]	84	[null]	[null]
6	AGO	Angola	Africa	33933611	24	5819	[null]	61	27	[null]
7	AIA	Anguilla	North America	15125	[null]	[null]	[null]	82	[null]	[null]
8	ATG	Antigua and Barbuda	North America	98728	232	21491	[null]	77	[null]	4
9	ARG	Argentina	South America	45605823	16	18934	1	77	[null]	5
10	ARM	Armenia	Asia	2968128	103	8788	2	75	94	4
11	ABW	Aruba	North America	107195	585	35974	[null]	76	[null]	[null]
12	OWID_ASI	Asia	[null]	4678444992	[null]	[null]	[null]	[null]	[null]	[null]
13	AUS	Australia	Oceania	25788217	3	44649	1	83	[null]	4
14	AUT	Austria	Europe	9043072	107	45437	1	82	[null]	7

Data Wrangling

For Single Time Point Data:

1. Steps:
2. Load Data on Jupiter notebook
3. Cleaned the data and created the data frames with the most recent data
   - `max_date = df.groupby('country_name')['date'].max()`
4. Created json files for the Data :
   - `df1.to_json('./export.json', orient='records')`

For Time Dependent Data:

Steps:

1. Load Data on Jupiter notebook
2. Selected columns of interest
3. Created json files for the Data :
   - `df1.to_json('./export.json', orient='records')`

Update GeoJSON with Properties from Data

DATA HUB

Country Polygons as GeoJSON

Certified

core

1. Get basic country GeoJSON from https://datahub.io/core/geo-countries

```
geo = pd.DataFrame(features)
geo.head()
```

	type	properties	geometry
0	Feature	{'ADMIN': 'Aruba', 'ISO_A3': 'ABW'}	{'type': 'Polygon', 'coordinates': [[[-69.9969...
1	Feature	{'ADMIN': 'Afghanistan', 'ISO_A3': 'AFG'}	{'type': 'Polygon', 'coordinates': [[[71.04980...
2	Feature	{'ADMIN': 'Angola', 'ISO_A3': 'AGO'}	{'type': 'MultiPolygon', 'coordinates': [[[[11...
3	Feature	{'ADMIN': 'Anguilla', 'ISO_A3': 'AIA'}	{'type': 'MultiPolygon', 'coordinates': [[[[-6...
4	Feature	{'ADMIN': 'Albania', 'ISO_A3': 'ALB'}	{'type': 'Polygon', 'coordinates': [[[19.74776...

2. Modify properties with COVID-19 data

```
geo['properties_new'] = properties_new
geo.head()
```

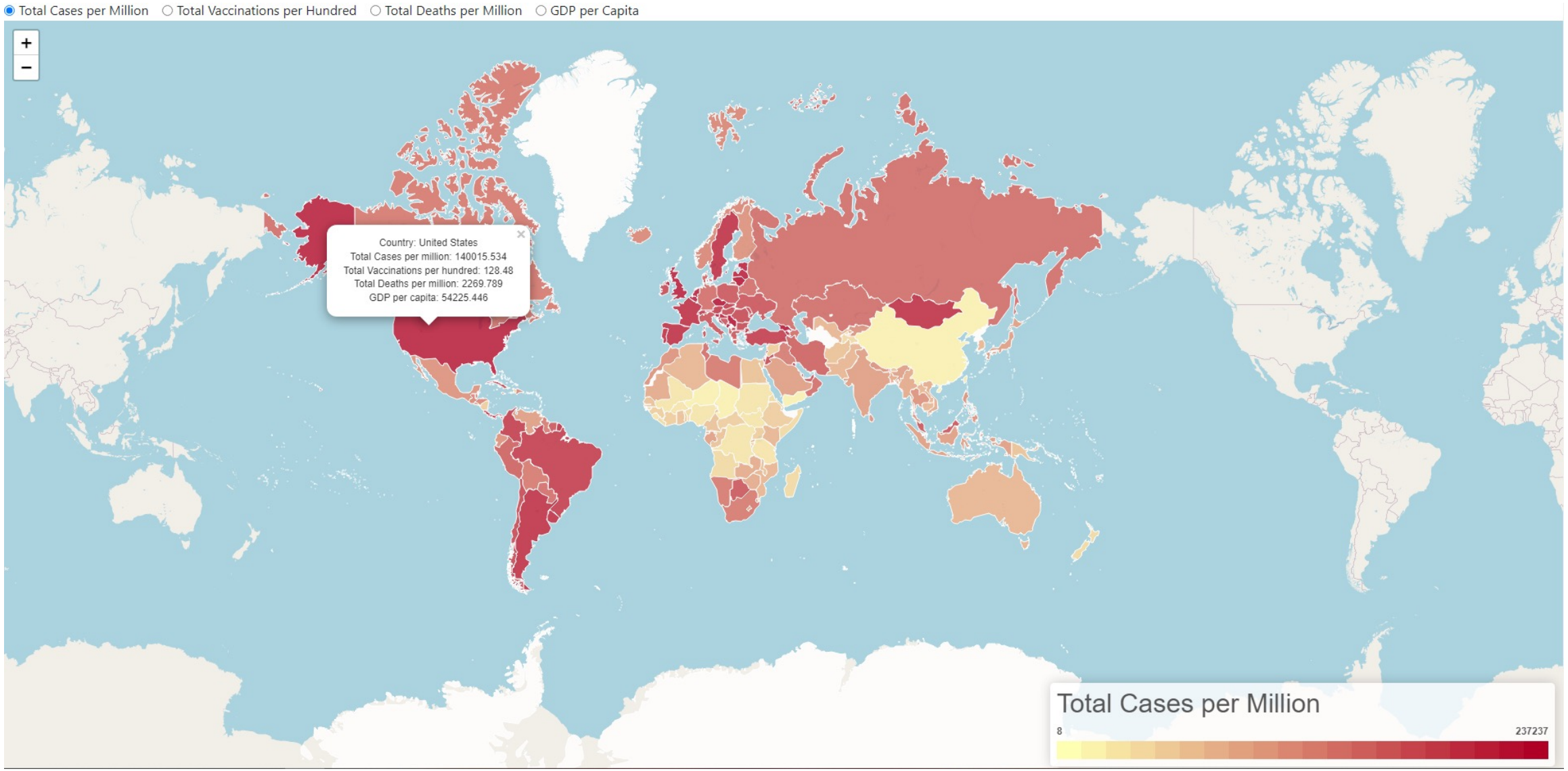
	type	properties	geometry	properties_new
0	Feature	{'ADMIN': 'Aruba', 'ISO_A3': 'ABW'}	{'type': 'Polygon', 'coordinates': [[[-69.9969...	{'iso_code': 'ABW', 'country_name': 'Aruba', '...
1	Feature	{'ADMIN': 'Afghanistan', 'ISO_A3': 'AFG'}	{'type': 'Polygon', 'coordinates': [[[71.04980...	{'iso_code': 'AFG', 'country_name': 'Afghanist...
2	Feature	{'ADMIN': 'Angola', 'ISO_A3': 'AGO'}	{'type': 'MultiPolygon', 'coordinates': [[[[11...	{'iso_code': 'AGO', 'country_name': 'Angola', ...
3	Feature	{'ADMIN': 'Anguilla', 'ISO_A3': 'AIA'}	{'type': 'MultiPolygon', 'coordinates': [[[[-6...	{'iso_code': 'AIA', 'country_name': 'Anguilla'...
4	Feature	{'ADMIN': 'Albania', 'ISO_A3': 'ALB'}	{'type': 'Polygon', 'coordinates': [[[19.74776...	{'iso_code': 'ALB', 'country_name': 'Albania',...

Build Interactive Choropleth Map

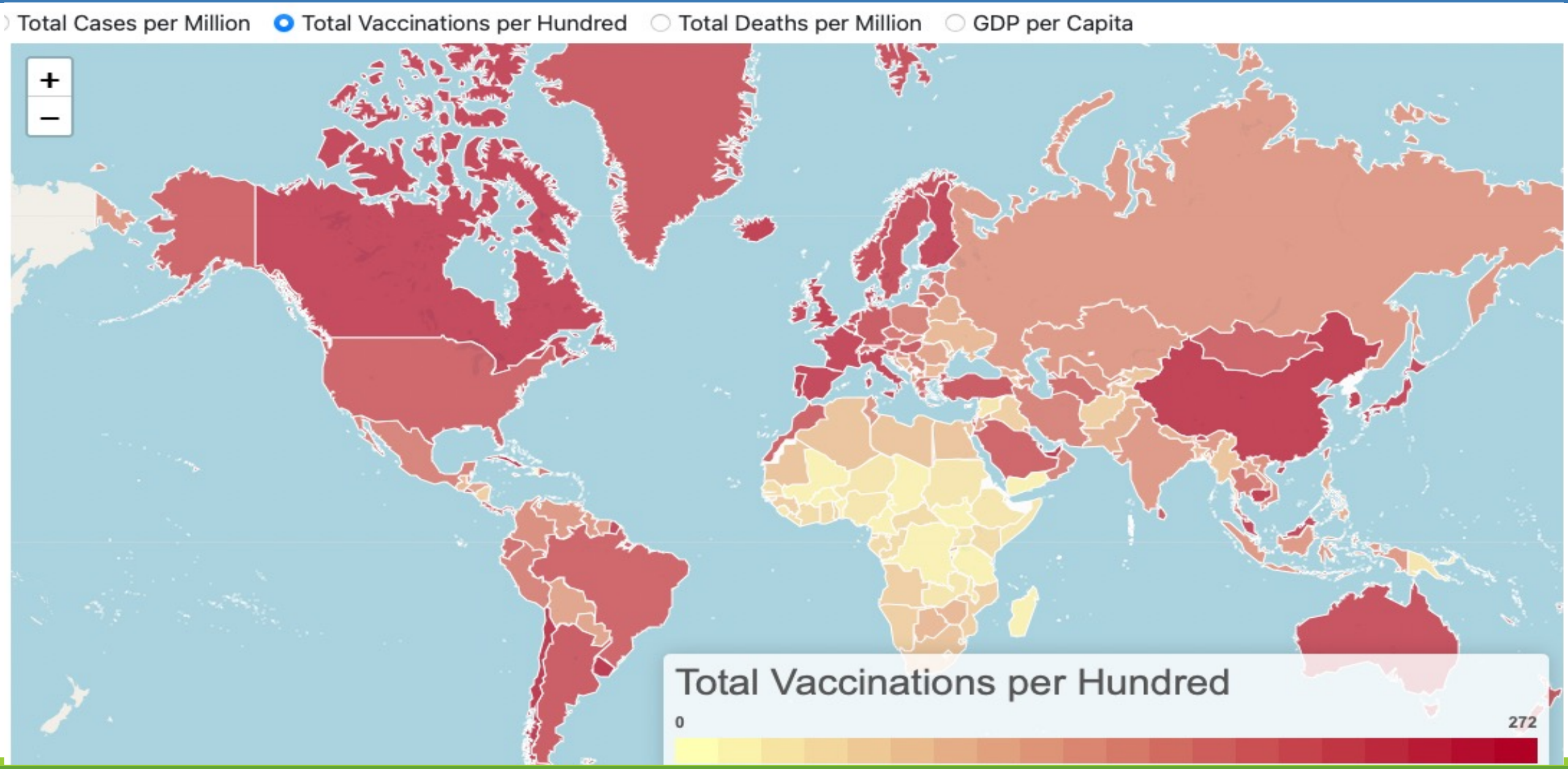
```
function makeChoropleth(data) {
  myMap.removeLayer(geojson)
// Create a new choropleth layer.
  geojson = L.choropleth(data, {

  // Define which property in the eatures to use.
  valueProperty: checkButton()[0],

  // Set the color scale.
  scale: ["#ffffb2", "#b10026"],

  // The number of breaks in the step range
  steps: 20,

  // q for quartile, e for equidistant, k for k-means
  mode: "q",
  style: {
    // Border color
    color: "#fff",
    weight: 1,
    fillOpacity: 0.8
  },

  // Binding a popup to each layer
  onEachFeature: function(feature, layer) {
    layer.bindPopup("Country: " + feature.properties.country_name +
        "<br>Total Cases per million: " + feature.properties.total_cases_per_million +
        "<br>Total Vaccinations per hundred: " + feature.properties.total_vaccinations_per_hundred +
        "<br>Total Deaths per million: " + feature.properties.total_deaths_per_million +
        "<br>GDP per capita: " + feature.properties.gdp_per_capita);
  }
}).addTo(myMap);
console.log('making map')
```

```
function checkButton() {
  if(document.getElementById('Total Cases per Million').checked) {
    selected = [document.getElementById('Total Cases per Million').value, document.getElementById('Total Cases per Million').id];
    return selected
  }
  else if(document.getElementById('Total Vaccinations per Hundred').checked) {
    selected = [document.getElementById('Total Vaccinations per Hundred').value, document.getElementById('Total Vaccinations per Hundred').id];
    return selected
  }
  else if(document.getElementById('Total Deaths per Million').checked) {
    selected = [document.getElementById('Total Deaths per Million').value, document.getElementById('Total Deaths per Million').id];
    return selected
  }
  else if(document.getElementById('GDP per Capita').checked) {
    selected = [document.getElementById('GDP per Capita').value, document.getElementById('GDP per Capita').id];
    return selected
  }
  else {
    selected = [document.getElementById('Total Cases per Million').value, document.getElementById('Total Cases per Million').id];
    return selected
  }
}
```

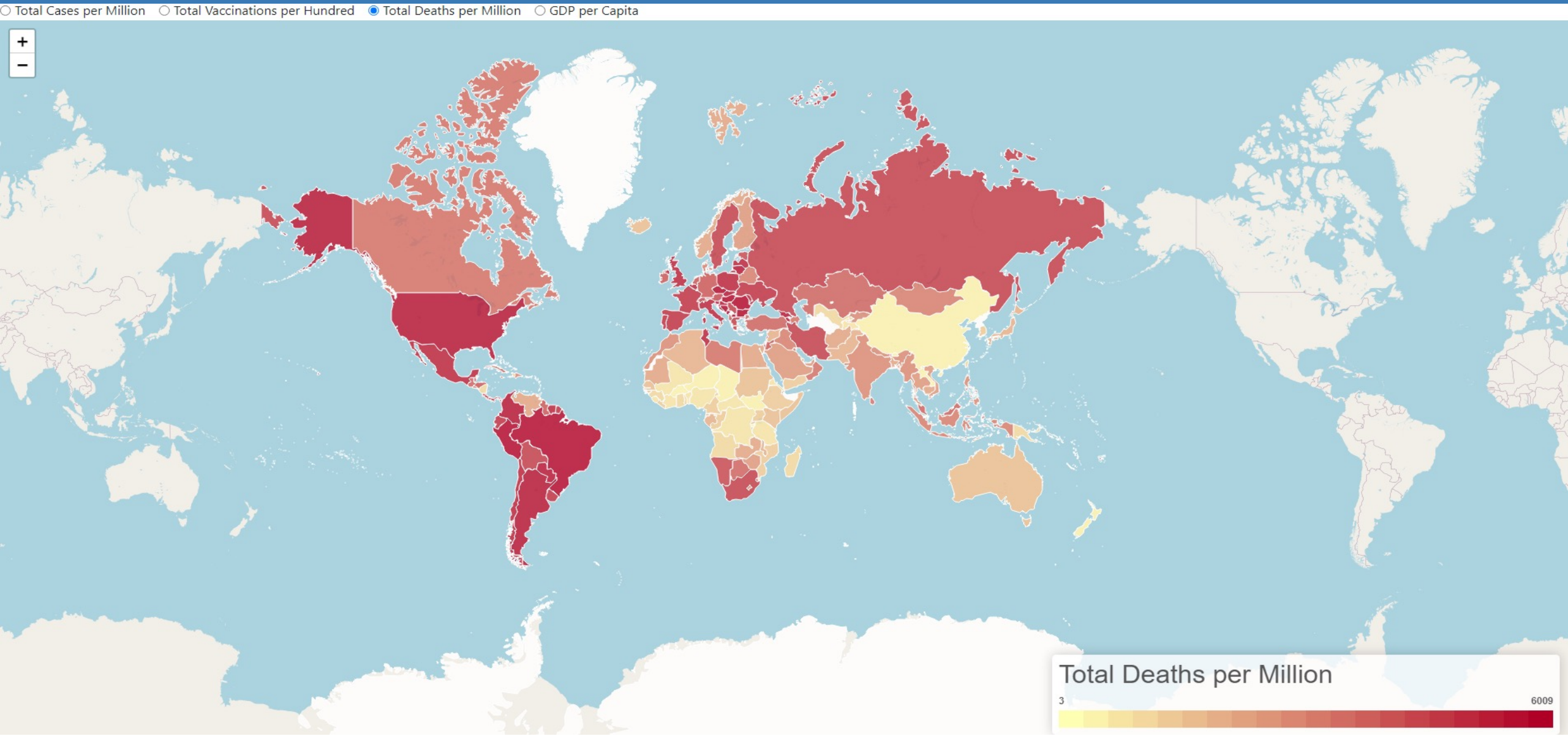
A choropleth map – Total Cases per million



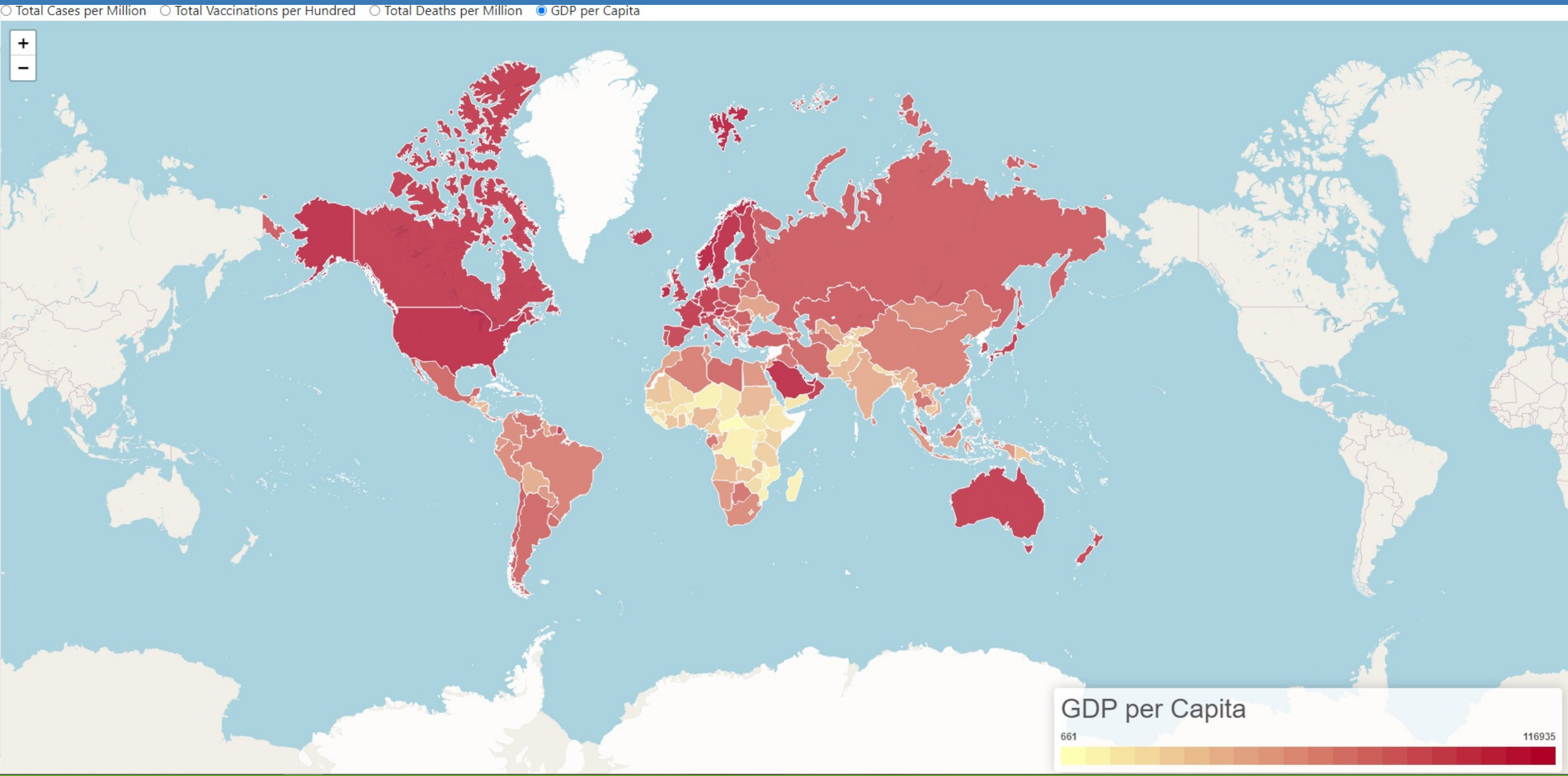
A choropleth map – Total vaccination per hundred



A choropleth map – Total Deaths per Million



A choropleth map – GDP per Capita

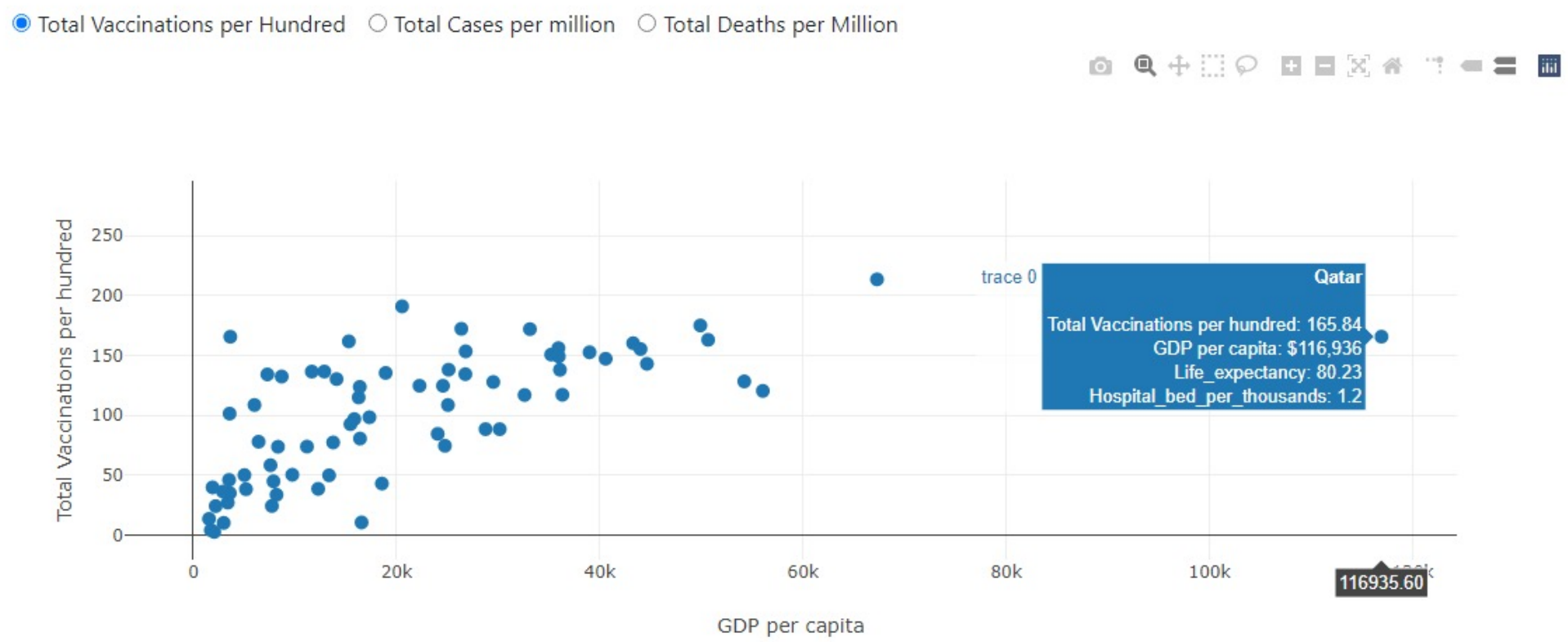


Interactive Scatter Plot - Code

```
function init() {
var trace1 = {
    x: gdp_list,
    y: checkButton()[0],

    mode: 'markers',
    marker: {size:10},
    text:text_list,

    hovertemplate:
                "<b>%{text.country}</b><br><br>" +
                "%{yaxis.title.text}: %{y:.0}<br>" +
                "%{xaxis.title.text}: %{x:$,.0f}<br>"+
                "Life_expectancy: %{text.expectency}<br>"+
                "Hospital_bed_per_thousands: %{text.hospital}<br>"


  };
```

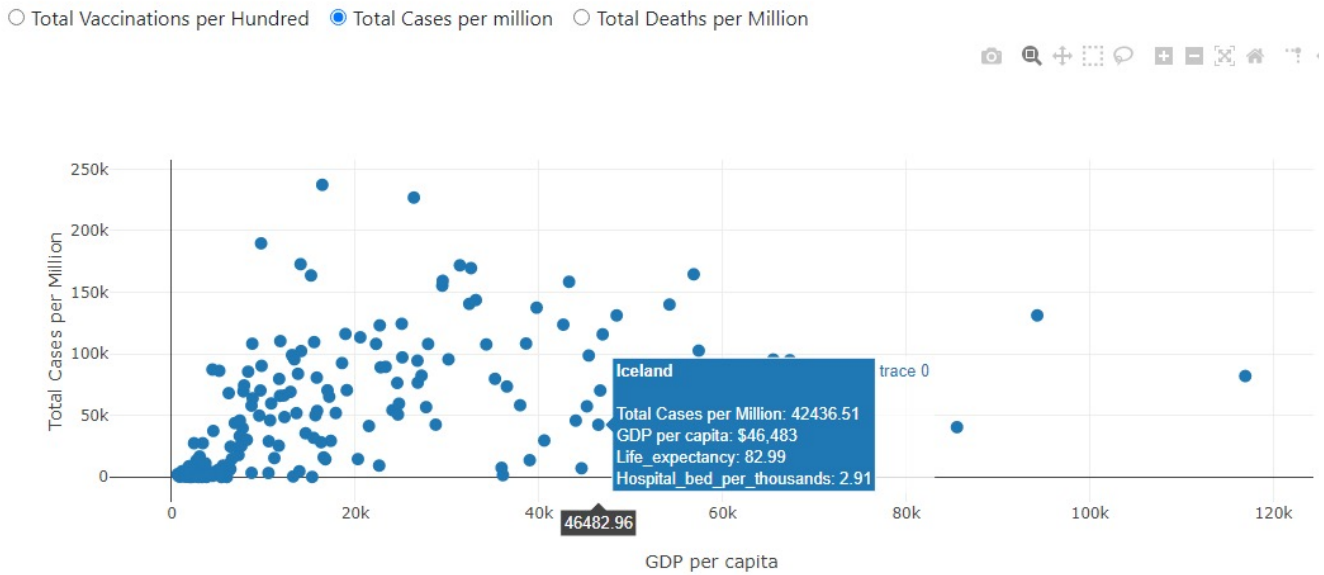
```
  var layout = {
    showlegend: false,
    height: Math.max(checkButton()),
    width: Math.max(checkButton()),
    xaxis:{ hoverformat: '.2f', title: 'GDP per capita'},
    yaxis:{ hoverformat: '.2r', title: checkButton()[1]},
  };

  Plotly.newPlot('bubble', data, layout);
}

function rand() {
  console.log("Hi!")
}

d3.selectAll("input").on("change", updatePlotly);
```

Interactive Scatter Plot - Total Vaccination

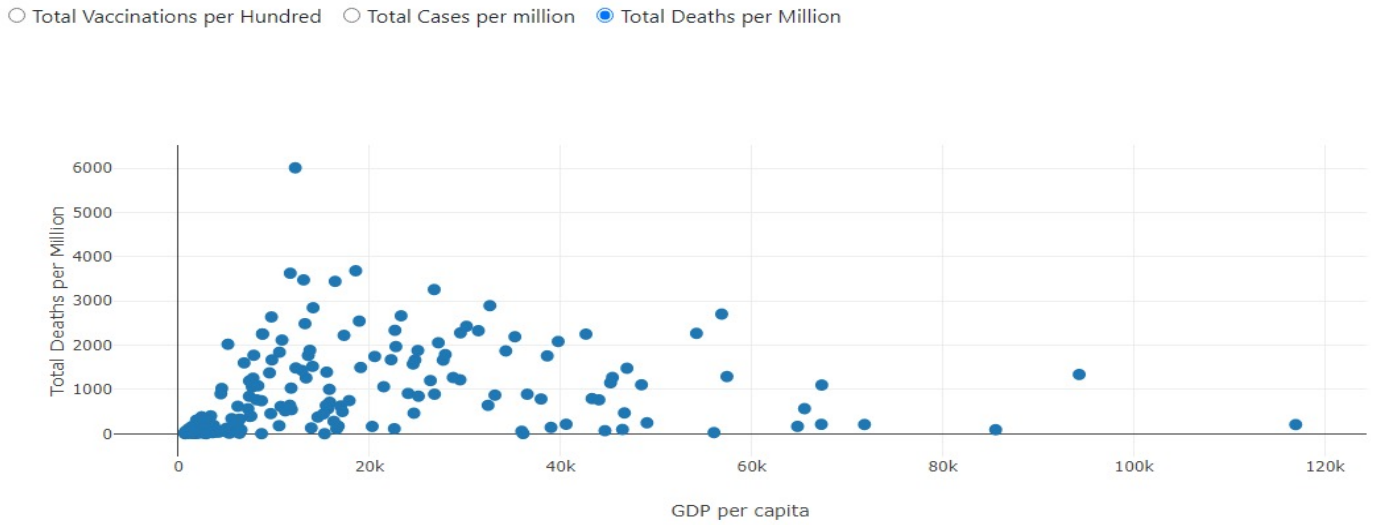


Countries with higher GDP have the higher number of total vaccination/100

Interactive Scatter Plot- Total Cases per million & total deaths



Countries with lower GDPs have the higher total cases and total deaths



Filtering By Continent

```
continent_list = data_owid_df.continent.unique()
print(continent_list)
continents = data_owid_df[data_owid_df.country_name.isin(continent_list)]
final_data = continents[['iso_code', 'country_name', 'continent', 'date', 'total_cases_per_million',
'total_deaths_per_million', 'total_vaccinations_per_hundred', 'new_cases', 'new_cases_smoothed',
'new_vaccinations', 'new_vaccinations_smoothed', 'new_vaccinations_smoothed_per_million',
'new_cases_smoothed_per_million', 'stringency_index']]
final_data.head()
```

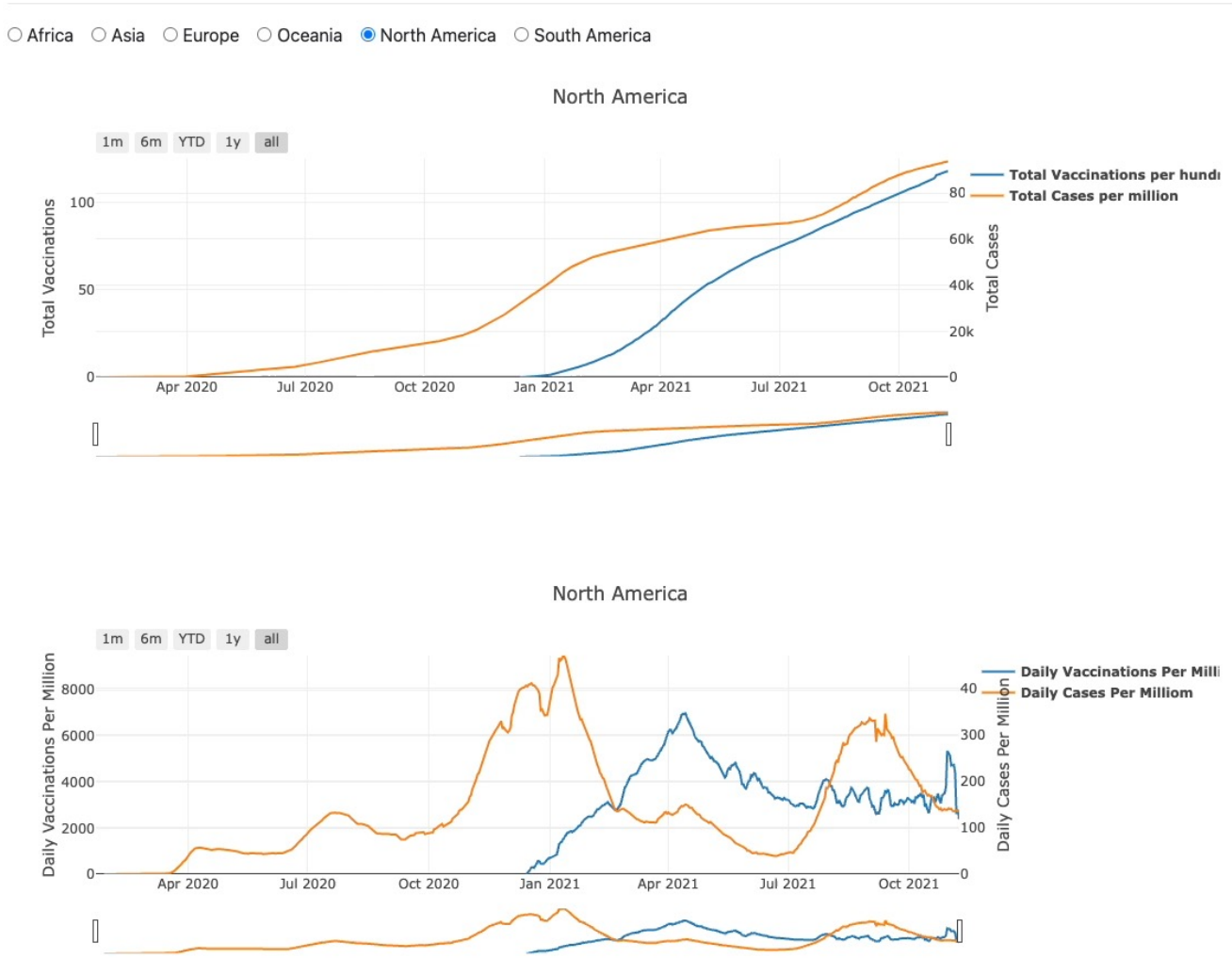
Creating the Graphs

```
function makeTotal() {
var continent = checkContinent();
const date_list =[];
const total_vaccination_list =[];
const total_cases_list =[];

data.forEach(function(value, i) {
    if (value.country_name == continent) {
        date_list[i] = value.date;
        total_vaccination_list[i] = value.total_vaccinations_per_hundred
        total_cases_list[i] = value.total_cases_per_million
    }
});

console.log(date_list)
console.log(total_vaccination_list)
```

```
let trace1 = {
  x: date_list,
  y: total_vaccination_list,
  type: 'scatter',
  name: 'Total Vaccinations per hundred',
};

let trace2 = {
  x: date_list,
  y: total_cases_list,
  yaxis: 'y2',
  type: 'scatter',
  name: 'Total Cases per million',
};

let plot_data = [trace1,trace2];

let layout = {
  title: checkContinent(),
  xaxis: {rangeselector: selectorOptions,
          rangeslider: {}},
  yaxis: {title: "Total Vaccinations", range: [0, 125]},
  yaxis2: {title: "Total Cases", range: [0, 95000],
           overlaying: "y",
           side: "right"},
};

Plotly.newPlot("scatter", plot_data, layout);
}
```

Vaccinations and Cases

```
var selectorOptions = {
  buttons: [{
      step: 'month',
      stepmode: 'backward',
      count: 1,
      label: '1m'
  }, {
      step: 'month',
      stepmode: 'backward',
      count: 6,
      label: '6m'
  }, {
      step: 'year',
      stepmode: 'todate',
      count: 1,
      label: 'YTD'
  }, {
      step: 'year',
      stepmode: 'backward',
      count: 1,
      label: '1y'
  }, {
      step: 'all',
  }],
};
```




thank you!