DATA1002 Project - Stage 1

Group 12

Members: 510615460, 480385312, 510465739, 500522378

[**1.0 Topic**](#_1p32mhmnv6yo) **1**

[**2.0 Datasets**](#_jmaaw5azl0hj) **2**

[2.1 COVID-19 Vaccination Progress](#_i7m8rcmqezyw) 2

[2.1.1 Exploration and Metadata](#_nnuch1pgukdj) 2

[2.1.2 Data Quality](#_je6xhjiq06va) 4

[2.1.3 Preliminary Analysis](#_3dpynkzifkw2) 5

[2.2 WHO-COVID-19](#_aaeky4ie6yyr) 12

[2.2.1 Exploration and Metadata](#_9ucqw1q97gm) 12

[2.2.2 Data Quality](#_x29ot6cs3gjs) 13

[2.2.3 Preliminary Analysis](#_a4x5zgakieei) 16

[2.3 Gross Domestic Product (GDP) - World Bank](#_b15mgrkm7hnq) 22

[2.3.1 Exploration and Metadata](#_psqg73dlqmct) 22

[2.3.2 Data Quality](#_jfxqna6sjbgl) 23

[2.3.3 Preliminary Analysis](#_ms76gfq0ekjk) 25

[2.4 COVID-19 Tests](#_a1p0ci9phjcm) 28

[2.4.1 Exploration and Metadata](#_8cpe0q7yxjkm) 28

[2.4.2 Data Quality](#_2s7886lgl4xb) 29

[2.4.3 Preliminary Analysis](#_g4jn15djexk0) 30

[**3.0 Data Integration**](#_ynr2wfoah53n) **34**

# 1.0 Topic

As a group, our topic was Health, so we agreed to explore datasets that are associated with the COVID-19 pandemic. The COVID-19 pandemic has had an adverse effect on people across the globe, so we believe it is beneficial to look at trends in the infections and vaccinations, especially seeing how they are affected by factors such as a country’s wealth or population.

While the information on the COVID-19 vaccination can be useful to anyone, there are some particular groups of stakeholders that may find our dataset and analysis useful. Public health officials would definitely be assisted by the tracking of COVID-19 vaccinations to identify how many vaccines have been administered, in order to see areas in which they could make improvements to the vaccine rollout. Another stakeholder that may be interested would be vaccine companies such as Pfizer or Moderna. These companies may be interested in the GDP of a country to determine worthwhile deals - for instance, a country with a higher GDP is typically more likely to have the medical infrastructure for a successful rollout. Of course, these companies may also be interested to see how many other vaccines are already available in a region and compare this to how much of the region’s population is awaiting a vaccine. Government officials would most probably take into account the number of COVID-19 cases and deaths to figure out whether a country or state is making progress in combating the virus, and this could help advise them for future actions, such as instigating stricter lockdowns or importing additional vaccine supplies.

# 2.0 Datasets

| **Dataset** | **SID** |
| --- | --- |
| [2.1 COVID-19 Vaccination Progress](#_i7m8rcmqezyw) | 510615460 |
| [2.2 WHO-COVID-19](#_aaeky4ie6yyr) | 480385312 |
| [2.3 GDP World Bank](#_b15mgrkm7hnq) | 510465739 |
| [2.4 COVID-19 Tests](#_bckihk1wwmra) | 500522378 |

## 2.1 COVID-19 Vaccination Progress

### 2.1.1 Exploration and Metadata

**Overview**

The *Our World in Data* COVID Vaccination dataset is a dataset that contains multiple information about the COVID-19 Vaccination collected by *Our World in Data*. While the *Our World in Data* offers a wide range of datasets, only one specific dataset is relevant and is being used. The dataset consists of a daily report of COVID vaccinations administered (as well as an averaged amount) in addition to the vaccine brands that were used in the corresponding countries for the period of December 2020 - September 2021.

Each row in the dataset consists of a report for a given date for a specific country. The format of the dataset is described in the following table. Note that we have decided to remove several of the columns in the dataset due to poor quality (discussed below). These columns will not be included in the table.

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| country | String | The country, territory, or area that the data relates to. |
| iso\_code | String | Country code in ISO Alpha-3 format. |
| date | Date | The date vaccines were administered |
| daily\_vaccinations | Integer | The number of vaccinations administered in a day |
| vaccines | String | The brands of vaccines available in each country |
| source\_name | String | Original source of each data |
| source\_website | URL | Original source of each data and the link to source\_name website |

The following columns will not be used in both the preliminary analysis and the integrated data as they were low-quality data (this will be explained further in the data quality section) :

* total\_vaccinations
* people \_vaccinated
* people\_fully\_vaccinated
* daily\_vaccinations\_raw
* total\_vaccinations\_per\_hundred
* people\_vaccinated\_per\_hundred
* people\_fully\_vaccinated\_per\_hundred
* daily\_vaccinations\_per\_million

**Data Provenance**

In accordance with the *Our World in Data* COVID vaccination [Statistics and Research](https://ourworldindata.org/covid-vaccinations), *Our World in Data* acquires its information from various sources depending on its country. At the top of the site, *Our World in Data* mentions that the information is derived from the most recent official numbers from governments and health ministries worldwide, such as the World Health Organization, a country’s Ministry of Health, etc. Additionally, *Our World in Data* explicitly states the original sources for each country in both the dataset (cited as a URL) and in the ‘source information country by country’ section at the very bottom of the website. This makes each available dataset a compilation of data from different sources.

In terms of restrictions, all data, visualizations, and code created by *Our World in Data* are open access to anyone. Permission to distribute and reproduce is not required. However, like any other, appropriate credit must be given by citing the source and indicating if changes were made. More information can be found on the [‘reuse our work freely’](https://ourworldindata.org/covid-vaccinations#licence) section of the website as well as the [Creative Commons BY license](https://creativecommons.org/licenses/by/4.0/).

**Strengths and Limitations**

The *Our World in Data* dataset has its fair share of strengths and limitations. At first glance, the dataset clearly had a large number of missing values. However, it is worth noting that these missing values are present because the values that can be found in these columns are averaged or summed values from daily reports.

Aside from that, the dataset had a few repeated values which may be a result of inconsistent reporting, underreporting, and differences in countries’ reporting methods. *Our World In Data* can be considered impartial and unbiased, but even so, some countries may have not been honest about their reports. Some countries also might not have the resources to report their vaccination progress as frequently as other countries, thus, choosing to report the same values for a few days. Lastly, a country may have also provided a certain amount of vaccines per day which justifies the repeated values.

As for the strength of the dataset, there was an adequate amount of information that can be used to analyze and compare. For instance, the dataset contains the number of vaccinations administered in a day and also the number of available vaccinations (vaccination brands) in a particular country.

### 2.1.2 Data Quality

**Inspection**

As mentioned, the chosen dataset appears to contain numerous missing values. Before cleaning the values, the following columns were manually deleted:

* iso\_code
* total\_vaccinations
* people \_vaccinated
* people\_fully\_vaccinated
* daily\_vaccinations\_raw
* total\_vaccinations\_per\_hundred
* people\_vaccinated\_per\_hundred
* people\_fully\_vaccinated\_per\_hundred
* daily\_vaccinations\_per\_million
* source\_name
* source\_website

These columns were removed as there were a large number of missing values within these columns because the values that were accounted for were averaged amounts from a certain period of time and were reported inconsistently. Thus, the dataset was left with 7 columns.

The Python script below was run to confirm that there were no missing values in the country, date or vaccines columns. We also check for missing values in the daily\_vaccinations column, in order to guide the cleaning process below.

**Inspection Code**

import csv

isFirstLine = True

csvReader = csv.reader(open("country\_vaccinations.csv"))

contents = list(csvReader)

total\_missing\_values = 0

num\_of\_values = 0

for row in contents:

if isFirstLine:

isFirstLine = False

else:

num\_of\_values += 1

country = row[0]

date = row[2]

vaccines = [12]

daily\_vaccinations = row[7]

if daily\_vaccinations == "":

total\_missing\_values += 1

if country == "" or date == "" or vaccines == "":

print("Found an unexpected missing value.")

percentage\_missing = (total\_missing\_values/num\_of\_values)\*100

print(total\_missing\_values, "out of", num\_of\_values, "values missing (" + str(round(percentage\_missing, 2)) + "%)")

**Output**

237 out of 42795 values missing (0.55%)

**Cleaning**

After identifying empty cells, the rest of the cleaning was done on Python to replace empty cells and negative values in the same column to 0 (which can be alternatively done using the *Find and Replace* window on Google Sheets). This was performed using the following Python script (which was then converted back into a csv file):

**Cleaning Code**

import csv

isFirstLine = True

csvReader = csv.reader(open("country\_vaccinations.csv"))

contents = list(csvReader)

for row in contents:

if isFirstLine:

isFirstLine = False

else:

#Replace missing values in the daily\_vaccinations column to 0

daily\_vaccinations = row[7]

if daily\_vaccinations == " ":

row[7] = '0'

#Replace any negative values to 0 in the daily\_vaccinations if there are any

if daily\_vaccinations < '0':

row[7] = '0'

writer = csv.writer(open("country-vaccinations-cleaned.csv", 'w', newline=''))

writer.writerows(contents)

COUNTA() was also used in a new column which was then deleted to compare if the country, date, daily\_ vaccinations, and vaccines (vaccine brands) columns had the same amount of cells.

### 2.1.3 Preliminary Analysis

For initial analysis of this dataset, the following values were determined **:**

* The country with the highest sum of vaccinations.
* The average number of vaccinations administered for each country.
* A list of countries that started vaccination in the year 2020.

**Summary Code**

#creating the dictionaries to save the keys and values

average\_vaccination\_per\_country = {}

country\_total\_vaccination = {}

country\_2020 = {}

is\_first\_line = True

for row in open("country-vaccinations-cleaned.csv"):

#skip the first line in the loop

if is\_first\_line:

is\_first\_line = False

#assign the columns to the variables

else:

values = row.split(",")

country = str(values[0])

date = str(values[2])

daily\_vaccination = float(values[3])

vaccines = str(values[4])

#setting up the ‘country\_total\_vaccination’ dictionary with country as the key and daily vaccination as the value

if country not in country\_total\_vaccination:

country\_total\_vaccination[country] = daily\_vaccination

else:

country\_total\_vaccination[country] += daily\_vaccination

#setting up the ‘average\_vaccination\_per\_country’ dictionary with country as the key and daily vaccination as the value

if country not in average\_vaccination\_per\_country:

average\_vaccination\_per\_country[country] = [daily\_vaccination]

else:

average\_vaccination\_per\_country[country].append(daily\_vaccination)

#setting up the ‘country\_2020 dictionary’ dictionary with country as the key and date as the value

if date.startswith('2020'):

if country not in country\_2020:

country\_2020[country] = [date]

else:

country\_2020[country].append(date)

#printing the country with the highest total number of vaccinations

max\_key = max(country\_total\_vaccination, key=country\_total\_vaccination.get)

print("\nThe country with the highest number of vaccination is " + max\_key)

#printing the average number of vaccination for each country

print("\nAverage number of vaccination for each country:")

for country in sorted(average\_vaccination\_per\_country):

vaccination = average\_vaccination\_per\_country[country]

average = sum(vaccination) / len(vaccination)

print("\t{}: {}".format(country, average))

#printing the countries that have started vaccinating since 2020

print("\nCountries that started vaccinating since 2020:")

for country in sorted(country\_2020):

print("\t" + country)

**Output**

The country with the highest number of vaccination is China

Average number of vaccination for each country:

Afghanistan: 9274.659685863875

Albania: 6190.063559322034

Algeria: 33034.69863013698

Andorra: 427.036866359447

Angola: 10724.72972972973

Anguilla: 89.12264150943396

Antigua and Barbuda: 388.92893401015226

Argentina: 173100.21031746033

Armenia: 1833.1761006289307

Aruba: 783.8765432098766

Australia: 99370.31862745098

Austria: 41351.75196850394

Azerbaijan: 30372.175965665236

Bahamas: 884.6136363636364

Bahrain: 9609.833333333334

Bangladesh: 121502.99107142857

Barbados: 1036.029702970297

Belarus: 11872.791836734694

Belgium: 64459.57142857143

Belize: 1280.4255319148936

Benin: 1076.3893805309735

Bermuda: 362.71729957805906

Bhutan: 8246.082278481013

Bolivia: 27384.27727272727

Bonaire Sint Eustatius and Saba: 0.0

Bosnia and Herzegovina: 4777.269841269841

Botswana: 3061.1614906832297

Brazil: 845869.8888888889

British Virgin Islands: 157.74528301886792

Brunei: 2129.3205128205127

Bulgaria: 9301.182539682539

Burkina Faso: 1183.3626373626373

Cambodia: 93264.38571428572

Cameroon: 2912.8322147651006

Canada: 200868.82397003745

Cape Verde: 1819.2095808383233

Cayman Islands: 404.64940239043824

Central African Republic: 1006.5283018867924

Chad: 626.3764705882353

Chile: 114281.140625

China: 7859932.060150376

Colombia: 178641.90452261307

Comoros: 1952.6279069767443

Congo: 2017.134328358209

Cook Islands: 198.2828282828283

Costa Rica: 16695.86

Cote d'Ivoire: 7385.494736842105

Croatia: 13118.668

Cuba: 117046.49541284403

Curacao: 1037.65

Cyprus: 4541.452282157677

Czechia: 45409.799212598424

Democratic Republic of Congo: 738.177304964539

Denmark: 32684.41825095057

Djibouti: 361.0289855072464

Dominica: 211.27317073170732

Dominican Republic: 55254.31034482759

Ecuador: 86630.84649122808

Egypt: 43125.99555555556

El Salvador: 32196.841584158417

England: 281271.2242647059

Equatorial Guinea: 2011.2196531791908

Estonia: 5052.708661417323

Eswatini: 1583.4026845637584

Ethiopia: 14279.375

Faeroe Islands: 310.2027649769585

Falkland Islands: 73.6268656716418

Fiji: 4812.428571428572

Finland: 27757.648

France: 350197.296

French Polynesia: 1061.0710900473935

Gabon: 851.4156626506024

Gambia: 2001.0058479532163

Georgia: 7247.242937853107

Germany: 404110.2834645669

Ghana: 9613.248226950354

Gibraltar: 346.51063829787233

Greece: 45408.98023715415

Greenland: 317.70403587443946

Grenada: 215.94634146341463

Guatemala: 24665.9792746114

Guernsey: 420.4159292035398

Guinea: 7205.981132075472

Guinea-Bissau: 254.91346153846155

Guyana: 2324.521951219512

Haiti: 754.36

Honduras: 22805.49738219895

Hong Kong: 39304.59390862944

Hungary: 46294.20948616601

Iceland: 2109.625

India: 2860690.940425532

Indonesia: 433409.35294117645

Iran: 133861.37619047618

Iraq: 18972.015789473684

Ireland: 27598.73092369478

Isle of Man: 544.5109170305677

Israel: 53650.93486590038

Italy: 309974.3385826772

Jamaica: 3117.0054945054944

Japan: 661125.5643564357

Jersey: 615.1627906976744

Jordan: 27234.878151260506

Kazakhstan: 56113.57534246575

Kenya: 15104.741935483871

Kiribati: 264.2413793103448

Kosovo: 5331.9876543209875

Kuwait: 12389.507936507936

Kyrgyzstan: 7200.300613496933

Laos: 24718.462427745664

Latvia: 5506.375451263538

Lebanon: 12118.271844660194

Lesotho: 763.8959537572255

Liberia: 763.2307692307693

Libya: 7926.166666666667

Liechtenstein: 167.24418604651163

Lithuania: 12613.94094488189

Luxembourg: 3045.764940239044

Macao: 2868.739336492891

Madagascar: 2376.635593220339

Malawi: 5219.764367816092

Malaysia: 183629.6717948718

Maldives: 3198.3981481481483

Mali: 2045.0621118012423

Malta: 3405.4008620689656

Mauritania: 1641.63125

Mauritius: 6980.559090909091

Mexico: 336528.97265625

Moldova: 6954.758064516129

Monaco: 197.33198380566802

Mongolia: 22060.035532994923

Montenegro: 2015.9095477386934

Montserrat: 14.177884615384615

Morocco: 155828.07657657657

Mozambique: 11975.866279069767

Myanmar: 29598.87019230769

Namibia: 1897.9464285714287

Nauru: 111.7925925925926

Nepal: 45918.482142857145

Netherlands: 91925.1129707113

New Caledonia: 703.3080568720379

New Zealand: 18653.775

Nicaragua: 3549.5401069518716

Niger: 3189.311688311688

Nigeria: 25089.510752688173

Niue: 51.22

North Macedonia: 6319.577114427861

Northern Cyprus: 1398.4532710280373

Northern Ireland: 9205.071161048689

Norway: 25532.61510791367

Oman: 15465.182539682539

Pakistan: 276849.5944700461

Palestine: 7834.651612903226

Panama: 19737.834782608697

Papua New Guinea: 843.4285714285714

Paraguay: 20809.45918367347

Peru: 91116.48803827752

Philippines: 182961.20942408376

Pitcairn: 0.7755102040816326

Poland: 144503.44444444444

Portugal: 58695.14566929134

Qatar: 17379.980694980695

Romania: 38712.27272727273

Russia: 308025.515037594

Rwanda: 11931.671568627451

Saint Helena: 83.8913043478261

Saint Kitts and Nevis: 241.88659793814432

Saint Lucia: 326.6326530612245

Saint Vincent and the Grenadines: 131.51162790697674

Samoa: 1044.2416666666666

San Marino: 246.21465968586386

Sao Tome and Principe: 283.9488636363636

Saudi Arabia: 154142.1024590164

Scotland: 28803.297794117647

Senegal: 9021.1875

Serbia: 24668.165975103733

Seychelles: 679.3521126760563

Sierra Leone: 1150.8502994011976

Singapore: 37516.487394957985

Sint Maarten (Dutch part): 202.59166666666667

Slovakia: 18613.89837398374

Slovenia: 7522.8031496062995

Solomon Islands: 492.48765432098764

Somalia: 1292.4825174825176

South Africa: 64266.44059405941

South Korea: 237222.51813471504

South Sudan: 380.61290322580646

Spain: 272108.22040816327

Sri Lanka: 95934.81081081081

Sudan: 6336.885542168675

Suriname: 1760.3299492385786

Sweden: 50294.613545816734

Switzerland: 37470.717054263565

Syria: 2177.1885714285713

Taiwan: 65860.35882352942

Tajikistan: 21652.880341880344

Tanzania: 10461.52380952381

Thailand: 162464.77297297298

Timor: 4118.371428571429

Togo: 3527.795580110497

Tokelau: 18.333333333333332

Tonga: 464.05223880597015

Trinidad and Tobago: 4526.176470588235

Tunisia: 32951.87640449438

Turkey: 408669.89873417723

Turkmenistan: 0.0

Turks and Caicos Islands: 217.64556962025316

Tuvalu: 62.13934426229508

Uganda: 7707.359550561798

Ukraine: 49282.4

United Arab Emirates: 71821.4693877551

United Kingdom: 342580.9588014981

United States: 1433515.5135135136

Uruguay: 29812.536458333332

Uzbekistan: 89649.05882352941

Vanuatu: 456.84444444444443

Venezuela: 41372.546875

Vietnam: 108917.65363128492

Wales: 17012.868913857677

Wallis and Futuna: 57.28395061728395

Yemen: 3786.7875

Zambia: 4066.6573426573427

Zimbabwe: 21572.425

Countries that started vaccinating since 2020:

Argentina

Austria

Bahrain

Belarus

Belgium

Bulgaria

Canada

Cayman Islands

Chile

China

Costa Rica

Croatia

Czechia

Denmark

England

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Israel

Italy

Kuwait

Latvia

Liechtenstein

Lithuania

Luxembourg

Mexico

Monaco

Northern Ireland

Norway

Oman

Poland

Portugal

Qatar

Romania

Russia

Scotland

Slovenia

Sweden

Switzerland

United Kingdom

United States

Wales

## 2.2 WHO-COVID-19

### 2.2.1 Exploration and Metadata

**Overview**

The WHO-COVID-19 dataset is a dataset consisting of information about the COVID-19 virus, based on information reported to the World Health Organisation (WHO). While the WHO provides public access for a large range of COVID-19 data, the specific dataset being used consists of daily reports of case numbers and death numbers for the period of January 3rd, 2020 - September 3rd, 2021. Each row in the dataset consists of a report for a given date for a specific country. The format of the dataset is as follows:

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| Date\_reported | Date | Date that the data was reported to WHO. |
| Country\_code | String | Country code in ISO Alpha-2 format. |
| Country | String | The country, territory or area that the data relates to. |
| WHO\_region | String | One of the following, based on the regional office the WHO Member State belongs to:   * Regional Office for Africa (AFRO) * Regional Office for the Americas (AMRO) * Regional Office for South-East Asia (SEARO) * Regional Office for Europe (EURO) * Regional Office for the Eastern Mediterranean (EMRO) * Regional Office for the Western Pacific (WPRO) |
| New\_cases | Integer | Difference between Cumulative\_cases reported for this Date\_reported and the previous Date\_reported. |
| Cumulative\_cases | Integer | Confirmed cumulative case value reported to the WHO up to this Date\_reported. |
| New\_deaths | Integer | Difference between Cumulative\_deaths reported for this Date\_reported and the previous Date\_reported. |
| Cumulative\_deaths | Integer | Confirmed cumulative death value reported to the WHO up to this Date\_reported. |

**Data Provenance**

The reports contained within this dataset originate from their respective countries, with the WHO collating the data into a single dataset. According to the WHO’s [Coronavirus Explorer](https://worldhealthorg.shinyapps.io/covid/), reports up to March 21st, 2020 are from “official communications under the International Health Regulations (IHR, 2005), complemented by monitoring the official ministries of health websites and social media accounts”. Since that date, reports have come from “WHO region-specific dashboards… and/or aggregate count data reported to WHO headquarters daily”. While it is not explicitly stated, it can be assumed that the original collectors of this data are the health departments (or equivalents) of each country’s governing body.

The data is openly published to the public and free to download and use. Permission is not required for usage of the data, but users must be clear about not having any affiliation with the WHO. This information is provided in more detail on the [World Health Organisation Policy page](https://www.who.int/about/policies/publishing/copyright).

**Strengths and Limitations**

The WHO-COVID-19 dataset is very useful for analysis, due to its high overall data quality and general lack of bias. Of particular note, there are no evident missing values, allowing for direct comparison between every country in the dataset for every single recorded day. While the WHO can generally be considered impartial, it is not clear if each country was accurate or honest in their reports. For instance, it is possible that some countries did not have the infrastructure to get accurate counts of cases and deaths each day, instead choosing to report figures in batches of several days. Additionally, it should be noted that a countries’ ability to process test results will impact the latency of its reports. For example, if a country takes 2 days to process COVID-19 tests, then the case numbers it reports on a given day actually reflect the case numbers from 2 days ago. This latency will be different for each country, and will also be dependent on the inundation of a country’s medical testing facilities, which could vary throughout the pandemic. For the sake of our analysis, we are choosing to ignore these limitations, but we note them here for completeness.

### 2.2.2 Data Quality

**Inspection**

This dataset does not appear to suffer from any severe quality issues, such as missing items or values. There are no apparent missing items - each country in the Country column has data reported for the exact same set of dates in Date\_reported. There are also no missing values - each of the values in New\_cases, Cumulative\_cases, New\_deaths and Cumulative\_deaths were logged to confirm that they were present and numerical. It is worth noting that in the Country\_code column, there are no values for the “Other” Country. This is understandable, since this country does not have a corresponding country code.

The dataset was also checked for incorrect values and default values. In the above checks to confirm no missing values, the values in New\_cases, Cumulative\_cases, New\_deaths and Cumulative\_deaths were checked to ensure that they were both positive and integers. This is because, obviously, cases and deaths cannot be negative or fractional. While they are rare, there are some negative results in the New\_cases and New\_deaths columns. Based on the data dictionary, the reports from each country were fed into the Cumulative\_cases and Cumulative\_deaths columns, and the other columns were calculated depending on these. Also, it is entirely possible that the data contains default values - specifically, missing values in New\_cases, Cumulative\_cases, New\_deaths and Cumulative\_deaths being replaced with “0”. It is impossible to know whether or not a “0” value refers to a date where the value reported was “0” versus the case where a report was not made at all. For our analysis, we have decided to operate on the assumption that all “0” values are true reports of “0”, rather than default values, because of this unknowability.

**Inspection Code**

import csv

def isNumeric(value):

try:

int(value)

return True

except:

return False

isFirstLine = True

csvReader = csv.reader(open("WHO-COVID-19-global-data.csv"))

contents = list(csvReader)

totalRows = 0

countriesPerDate = {}

for row in contents:

if isFirstLine:

isFirstLine = False

else:

totalRows += 1

date = row[0]

country = row[2]

if date not in countriesPerDate:

countriesPerDate[date] = [country]

else:

countriesPerDate[date].append(country)

newCases = row[4]

cumulativeCases = row[5]

newDeaths = row[6]

cumulativeDeaths = row[7]

if newCases == " " or cumulativeCases == " " or newDeaths == " " or cumulativeDeaths == " ":

print("Found a missing value for a field.")

continue

if not isNumeric(newCases) or not isNumeric(cumulativeCases) or not isNumeric(newDeaths) or not isNumeric(cumulativeDeaths):

print("Found a non-numeric value for a numeric field.")

continue

if not float(newCases).is\_integer() or not float(cumulativeCases).is\_integer() or not float(newDeaths).is\_integer() or not float(cumulativeDeaths).is\_integer():

print("Found a non-integer value for a numeric field.")

continue

if int(newCases) < 0 or int(cumulativeCases) < 0 or int(newDeaths) < 0 or int(cumulativeDeaths) < 0:

print("Found a negative value for a numeric field.")

continue

expectedCountryLength = len(list(countriesPerDate.values())[0])

for countryList in countriesPerDate.values():

if len(countryList) != expectedCountryLength:

print("Found a date with a different number of countries.")

**Output**

Found a negative value for a numeric field.

Found a negative value for a numeric field.

[...]

Found a negative value for a numeric field.

Found a negative value for a numeric field.

**Cleaning**

In order to aid our analysis, some minor cleaning and column removal was performed on the dataset. One of the Country values - “occupied Palestinian territory, including east Jerusalem” - includes the “,” character, which can cause some issues during parsing of the file by our Python programs. This Country value was replaced with the value “occupied Palestinian territory including east Jerusalem”, removing the comma. The decision was made to remove the Cumulative\_cases and Cumulative\_deaths columns, since these could be easily recalculated using the New\_cases and New\_deaths columns. With this done, the negative values found in New\_cases and New\_deaths were replaced with values of “0”. When the cumulative totals of each of these fields are recalculated, the correction will take effect there as well. Additionally, the Country\_code and WHO\_region columns were removed, since they are not useful for our analysis. This cleaning and column removal was performed using the script below:

**Cleaning Code**

import csv

isFirstLine = True

csvReader = csv.reader(open("WHO-COVID-19-global-data.csv"))

contents = list(csvReader)

for row in contents:

if isFirstLine:

isFirstLine = False

# Remove columns for Cumulative\_cases, Cumulative\_deaths, WHO\_region and Country\_code.

row.pop(7)

row.pop(5)

row.pop(3)

row.pop(1)

else:

# Remove commas from the Country field.

country = row[2]

if "," in country:

row[2] = country.replace(",", "")

# Replace negative values in New\_cases.

newCases = int(row[4])

if newCases < 0:

row[4] = "0"

# Replace negative values in New\_deaths.

newDeaths = int(row[6])

if newDeaths < 0:

row[6] = "0"

# Remove columns for Cumulative\_cases, Cumulative\_deaths, WHO\_region and Country\_code.

row.pop(7)

row.pop(5)

row.pop(3)

row.pop(1)

writer = csv.writer(open("WHO-COVID-19-global-data-cleaned.csv", 'w', newline=''))

writer.writerows(contents)

### 2.2.3 Preliminary Analysis

For initial analysis of this dataset, the following summary values were determined:

* The country with the highest new deaths in a single day.
* The country with the lowest total cumulative deaths.
* The average number of new cases for each country.

The script below was run to determine these summaries:

**Summary Code**

import csv

isFirstLine = True

# Tuple of (deaths, country, date).

highestNewDeaths = None

# Dictionary of {country -> cumulativeDeaths}, where cumulativeDeaths is the highest cumulative death value for the country.

cumulativeDeathsPerCountry = {}

# Dictionary of {country -> [cases]}, where [cases] is an array of case values for each day for the country.

newCasesPerCountry = {}

csvReader = csv.reader(open("WHO-COVID-19-global-data-cleaned.csv"))

contents = list(csvReader)

for row in contents:

if isFirstLine:

isFirstLine = False

else:

date = row[0]

country = row[1]

newCases = int(row[2])

newDeaths = int(row[3])

# Determine if the new deaths value is higher than the current highest.

if not highestNewDeaths or newDeaths > highestNewDeaths[0]:

highestNewDeaths = (newDeaths, country, date)

# Update the cumulative deaths for the country if it has increased.

if country not in cumulativeDeathsPerCountry:

cumulativeDeathsPerCountry[country] = newDeaths

else:

cumulativeDeathsPerCountry[country] += newDeaths

# Keep a record of the number of new cases for the country.

if country not in newCasesPerCountry:

newCasesPerCountry[country] = [newCases]

else:

newCasesPerCountry[country].append(newCases)

print("Highest new deaths was {} in {} on {}.".format(highestNewDeaths[0], highestNewDeaths[1], highestNewDeaths[2]))

lowestCumulativeDeaths = None

for country, deaths in cumulativeDeathsPerCountry.items():

if not lowestCumulativeDeaths or deaths < lowestCumulativeDeaths[1]:

lowestCumulativeDeaths = (country, deaths)

print("Country with the lowest total cumulative deaths was {} with {} deaths.".format(lowestCumulativeDeaths[0], lowestCumulativeDeaths[1]))

print("Average new cases per day:")

for country in sorted(newCasesPerCountry):

allNewCases = newCasesPerCountry[country]

average = sum(allNewCases) / len(allNewCases)

print("\t{}: {}".format(country, average))

**Output**

Highest new deaths was 8786 in Ecuador on 2021-07-21.

Country with the lowest total cumulative deaths was American Samoa with 0 deaths.

Average new cases per day:

Afghanistan: 251.4344262295082

Albania: 242.98688524590165

Algeria: 322.8114754098361

American Samoa: 0.0

Andorra: 24.675409836065572

Angola: 78.69508196721311

Anguilla: 0.419672131147541

Antigua and Barbuda: 2.855737704918033

Argentina: 8509.750819672132

Armenia: 399.9688524590164

Aruba: 23.895081967213116

Australia: 92.72950819672131

Austria: 1126.622950819672

Azerbaijan: 709.0081967213115

Bahamas: 30.262295081967213

Bahrain: 446.94262295081967

Bangladesh: 2475.8737704918035

Barbados: 8.314754098360655

Belarus: 795.1918032786886

Belgium: 1950.344262295082

Belize: 27.037704918032787

Benin: 26.118032786885244

Bermuda: 4.849180327868853

Bhutan: 4.255737704918033

Bolivia (Plurinational State of): 805.4754098360655

Bonaire: 2.880327868852459

Bosnia and Herzegovina: 352.972131147541

Botswana: 257.2573770491803

Brazil: 34105.270491803276

British Virgin Islands: 4.2098360655737705

Brunei Darussalam: 4.852459016393443

Bulgaria: 752.5426229508197

Burkina Faso: 22.619672131147542

Burundi: 21.42622950819672

Cabo Verde: 58.4

Cambodia: 153.97704918032787

Cameroon: 138.04918032786884

Canada: 2463.9639344262296

Cayman Islands: 1.1295081967213114

Central African Republic: 18.518032786885247

Chad: 8.190163934426229

Chile: 2688.0295081967215

China: 201.8622950819672

Colombia: 8050.954098360656

Comoros: 6.675409836065573

Congo: 22.275409836065574

Cook Islands: 0.0

Costa Rica: 764.8754098360656

Croatia: 617.0770491803279

Cuba: 1091.2655737704918

CuraÃ§ao: 25.08032786885246

Cyprus: 187.48524590163933

Czechia: 2754.173770491803

CÃ´te dâ€™Ivoire: 92.65737704918033

Democratic People's Republic of Korea: 0.0

Democratic Republic of the Congo: 90.6672131147541

Denmark: 569.2

Djibouti: 19.28688524590164

Dominica: 2.685245901639344

Dominican Republic: 575.0639344262295

Ecuador: 823.1901639344262

Egypt: 473.3311475409836

El Salvador: 157.48688524590165

Equatorial Guinea: 15.536065573770491

Eritrea: 10.888524590163934

Estonia: 234.72295081967212

Eswatini: 71.8967213114754

Ethiopia: 509.8262295081967

Falkland Islands (Malvinas): 0.10491803278688525

Faroe Islands: 1.6426229508196721

Fiji: 77.46885245901639

Finland: 211.64754098360655

France: 10828.449180327869

French Guiana: 57.88360655737705

French Polynesia: 69.03114754098361

Gabon: 42.43934426229508

Gambia: 15.960655737704919

Georgia: 917.9393442622951

Germany: 6531.726229508196

Ghana: 196.72622950819672

Gibraltar: 8.798360655737705

Greece: 973.2262295081968

Greenland: 0.5688524590163935

Grenada: 1.0885245901639344

Guadeloupe: 74.41475409836066

Guam: 17.07049180327869

Guatemala: 779.5868852459016

Guernsey: 2.1295081967213116

Guinea: 48.532786885245905

Guinea-Bissau: 9.59672131147541

Guyana: 42.24590163934426

Haiti: 34.38852459016393

Holy See: 0.04262295081967213

Honduras: 559.8655737704918

Hungary: 1332.8524590163934

Iceland: 17.888524590163936

India: 53939.81803278688

Indonesia: 6749.0

Iran (Islamic Republic of): 8287.724590163934

Iraq: 3118.7

Ireland: 577.916393442623

Isle of Man: 10.973770491803279

Israel: 1801.1918032786884

Italy: 7464.329508196722

Jamaica: 113.20327868852459

Japan: 2499.4737704918034

Jersey: 15.327868852459016

Jordan: 1308.3459016393442

Kazakhstan: 1433.1295081967214

Kenya: 389.91967213114754

Kiribati: 0.0

Kosovo[1]: 240.81639344262294

Kuwait: 672.067213114754

Kyrgyzstan: 288.84098360655736

Lao People's Democratic Republic: 25.34262295081967

Latvia: 235.15737704918033

Lebanon: 988.9967213114754

Lesotho: 23.598360655737704

Liberia: 9.388524590163934

Libya: 511.6655737704918

Liechtenstein: 5.577049180327869

Lithuania: 493.716393442623

Luxembourg: 124.55901639344262

Madagascar: 70.28360655737706

Malawi: 99.44754098360656

Malaysia: 2927.8754098360655

Maldives: 133.5639344262295

Mali: 24.442622950819672

Malta: 59.544262295081964

Marshall Islands: 0.006557377049180328

Martinique: 62.27049180327869

Mauritania: 55.52295081967213

Mauritius: 18.32950819672131

Mayotte: 32.58196721311475

Mexico: 5524.175409836065

Micronesia (Federated States of): 0.0

Monaco: 5.278688524590164

Mongolia: 363.1901639344262

Montenegro: 191.0360655737705

Montserrat: 0.047540983606557376

Morocco: 1421.2590163934426

Mozambique: 241.09180327868853

Myanmar: 671.3262295081968

Namibia: 205.14098360655737

Nauru: 0.0

Nepal: 1257.8213114754099

Netherlands: 3191.2377049180327

New Caledonia: 0.22295081967213115

New Zealand: 5.527868852459016

Nicaragua: 15.218032786885246

Niger: 9.611475409836066

Nigeria: 317.4491803278689

Niue: 0.0

North Macedonia: 292.0639344262295

Northern Mariana Islands (Commonwealth of the): 0.3983606557377049

Norway: 267.8081967213115

Oman: 495.8459016393443

Other: 1.2524590163934426

Pakistan: 1914.4114754098362

Palau: 0.003278688524590164

Panama: 751.0770491803279

Papua New Guinea: 29.881967213114756

Paraguay: 751.8262295081968

Peru: 3526.245901639344

Philippines: 3312.268852459016

Pitcairn Islands: 0.0

Poland: 4737.332786885246

Portugal: 1708.7245901639344

Puerto Rico: 281.23770491803276

Qatar: 381.8196721311475

Republic of Korea: 421.4918032786885

Republic of Moldova: 440.80655737704916

Romania: 1808.52131147541

Russian Federation: 11434.711475409837

Rwanda: 145.38032786885245

RÃ©union: 82.5344262295082

Saba: 0.018032786885245903

Saint BarthÃ©lemy: 2.6098360655737705

Saint Helena: 0.0

Saint Kitts and Nevis: 1.881967213114754

Saint Lucia: 13.642622950819671

Saint Martin: 5.501639344262295

Saint Pierre and Miquelon: 0.04918032786885246

Saint Vincent and the Grenadines: 3.860655737704918

Samoa: 0.001639344262295082

San Marino: 8.762295081967213

Sao Tome and Principe: 4.3180327868852455

Saudi Arabia: 893.1327868852459

Senegal: 119.6344262295082

Serbia: 1261.972131147541

Seychelles: 33.16393442622951

Sierra Leone: 10.440983606557378

Singapore: 111.46065573770491

Sint Eustatius: 0.03934426229508197

Sint Maarten: 6.0967213114754095

Slovakia: 648.4131147540984

Slovenia: 440.3868852459016

Solomon Islands: 0.03278688524590164

Somalia: 28.632786885245903

South Africa: 4584.270491803279

South Sudan: 18.808196721311475

Spain: 7985.973770491803

Sri Lanka: 734.027868852459

Sudan: 62.10819672131147

Suriname: 48.52295081967213

Sweden: 1851.0344262295082

Switzerland: 1282.4491803278688

Syrian Arab Republic: 45.97540983606557

Tajikistan: 28.381967213114756

Thailand: 2046.295081967213

The United Kingdom: 11250.668852459017

Timor-Leste: 28.490163934426228

Togo: 36.032786885245905

Tokelau: 0.0

Tonga: 0.0

Trinidad and Tobago: 73.7

Tunisia: 1095.1655737704918

Turkey: 10550.447540983607

Turkmenistan: 0.0

Turks and Caicos Islands: 4.380327868852459

Tuvalu: 0.0

Uganda: 163.44426229508196

Ukraine: 3759.9032786885246

United Arab Emirates: 1180.8688524590164

United Republic of Tanzania: 2.240983606557377

United States Virgin Islands: 9.727868852459016

United States of America: 64426.67213114754

Uruguay: 631.2754098360656

Uzbekistan: 259.9360655737705

Vanuatu: 0.004918032786885246

Venezuela (Bolivarian Republic of): 550.9639344262295

Viet Nam: 797.9131147540984

Wallis and Futuna: 0.7442622950819672

Yemen: 13.034426229508197

Zambia: 338.8606557377049

Zimbabwe: 205.11147540983606

occupied Palestinian territory including east Jerusalem: 613.5360655737705

## 2.3 Gross Domestic Product (GDP) - World Bank

### 2.3.1 Exploration and Metadata

**Overview**

This data set was sourced from the World Bank and contains data on the gross domestic product (GDP) of 275 countries from 1960 to 2020. The data is a subsection of a larger data set containing information on various World Development Indicators for different countries.

The data is presented in a wide format, with each row containing the GDP from 1960 - 2020 for its respective country. The format of the data is as follows:

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| Country Name | String | Name of the country from which the data was collected. |
| Country Code | String | Country code in ISO Alpha-3 format. |
| Indicator Name | String | The name of the World Development Indicator being displayed (this data set is a subsection of a larger data set with other indicators). |
| Indicator code | String | The code for the World Development Indicator given by the World Bank. |
| 1960 - 2020 (Separate fields) | Float | The gross domestic product (GDP) in USD for the respective financial year for each country. |

**Data Provenance**

This data set was sourced from the World Bank at the URL: <https://data.worldbank.org/indicator/NY.GDP.MKTP.CD> on the 7th September 2021. According to the World Bank, the data came from “World Bank national accounts data, and OECD National Accounts data files” and was last updated on the 30th July 2021.

This data is publicly available and licenced under a Creative Commons Attribution 4.0 International License (CC BY 4.0) which allows for copying and redistributing material in any medium or format, and remixing, transforming and building upon the material for any purpose as long as proper attribution to the creator is given. More information can be found on [Terms of Use for Datasets](https://www.worldbank.org/en/about/legal/terms-of-use-for-datasets).

**Strengths and Limitations**

This data set has a number of strengths that make it good for analysis. One strength is that it has a large amount of data, containing the GDPs of 275 countries for a 60 year period. This will allow for a deeper and better quality analysis. Another strength is that it comes from a reputable source, The World Bank. This means that we can be confident in the accuracy of the data. One considerable limitation however is the large quantity of missing data. While some countries have data for the entire 60 year period, other countries have gaps with no GDP recorded. There could be many reasons for this, some may have to do with issues with data collection, however, they may also be political, for example, some countries may not have existed or been recognised as a country for different parts of the 60 year period.

### 2.3.2 Data Quality

The GDP dataset from the World Bank was of high quality, except for a large quantity of missing values. Upon inspection, almost 20% of the total values were missing. The following code was used to count the number of empty cells in the data, compare it to the overall number of values, and calculate the percentage of values that were missing.

**Inspection Code**

is\_first\_line = True

total\_missing\_values = 0

num\_of\_values = 0

for row in open("gdp\_data.txt"):

if is\_first\_line:

is\_first\_line = False

else:

values = row.split(",")

missing\_values = 0

for value in values:

num\_of\_values += 1

if value == "":

missing\_values += 1

total\_missing\_values += missing\_values

percentage\_missing = (total\_missing\_values/num\_of\_values)\*100

print(total\_missing\_values, "out of", num\_of\_values, "values missing (" + str(round(percentage\_missing, 2)) + "%)")

**Output**

3457 out of 17303 values missing (19.98%)

However, there were no other apparent data quality issues with this data set. The following code was used to check for default values and upon inspection there were none.

**Inspection Code**

import csv

gdp\_data = open("gdp\_data.csv")

csv\_reader = csv.reader(gdp\_data)

first\_line = True

second\_line = True

third\_line = True

fourth\_line = True

header\_line = True

# num\_of\_occurances\_of\_each\_value is a dictionary with each unique value as the key and the number of times they occur as the value.

num\_of\_occurances\_of\_each\_value = {}

for row in csv\_reader:

for value in row:

if value not in num\_of\_occurances\_of\_each\_value:

num\_of\_occurances\_of\_each\_value[value] = 1

else:

num\_of\_occurances\_of\_each\_value[value] += 1

# potential\_default\_values is a dictionary with any keys and values from num\_of\_occurances\_of\_each\_value that have a value more than 10.

potential\_default\_values = {}

for key in num\_of\_occurances\_of\_each\_value:

if num\_of\_occurances\_of\_each\_value[key] > 10:

potential\_default\_values[key] = num\_of\_occurances\_of\_each\_value[key]

print(potential\_default\_values)

**Output**

{'': 3769, 'GDP (current US$)': 266, 'NY.GDP.MKTP.CD': 266}

As shown above, the only values that occurred more than 10 times were empty values, which was previously found, and the indicator name and code which were the same in every row.

**Cleaning**

In order to fix these issues, rows with missing values needed to be removed. However, as we are only interested in the 2019 and 2020 GDPs for our analysis, the only rows that needed to be removed are those that have either 2019 or 2020 missing. Therefore, the columns with the GDPs for 1960-2018 were also removed. Furthermore, the country code, indicator and indicator code were also not relevant to our analysis so those columns were also removed. The original csv file also had 4 lines at the beginning that either contained some metadata that was not the schema or were blank, so these needed to be removed. The following code was used to read the csv file with the data, skip the first 4 rows then write each country and their 2019 and 2020 GDPs into a new csv file. In doing so, the code also checked whether the country had values for both 2019 and 2020, and if it did not, the row was not written into the new file, thus removing the missing data. When reading and writing the values for 2019 and 2020 GDP, the indexes -3 and -2 respectively had to be used as the original file had an extra empty value at the end of each line. Only writing in the 3rd and 2nd last values eliminating this extra missing data from the new csv file.

**Cleaning Code**

import csv

cleaned\_data = open("gdp-cleaned.csv", "w")

csv\_writer = csv.writer(cleaned\_data)

gdp\_data = open("gdp\_data.csv")

csv\_reader = csv.reader(gdp\_data)

first\_line = True

second\_line = True

third\_line = True

fourth\_line = True

header\_line = True

for row in csv\_reader:

# skipping the first 4 lines

if first\_line:

first\_line = False

elif second\_line:

second\_line = False

elif third\_line:

third\_line = False

elif fourth\_line:

fourth\_line = False

# Writing the header row

elif header\_line:

csv\_writer.writerow([row[0], row[-3], row[-2]])

header\_line = False

# Writing the values

else:

if row[-2] != "" and row[-3] != "":

csv\_writer.writerow([row[0], row[-3], row[-2]])

### 2.3.3 Preliminary Analysis

For initial analysis of this dataset, the following summary values were determined:

* The country with the highest GDP in 2020.
* The average GDP per year.
* The number of countries with a GDP above average in 2019.

The script below was run to determine these summaries:

**Summary 1 Code**

# Country with highest GDP in 2020

import csv

is\_first\_line = True

max\_gdp = 0

country\_with\_max\_gdp = ""

cleaned\_gdp = open("gdp-cleaned.csv")

csv\_reader = csv.reader(cleaned\_gdp)

for row in csv\_reader:

if is\_first\_line:

is\_first\_line = False

else:

country = row[0]

gdp\_2020 = float(row[2])

if gdp\_2020 > max\_gdp:

max\_gdp = gdp\_2020

country\_with\_max\_gdp = country

print("The country with the highest GDP in the 2020 FY is", country\_with\_max\_gdp, "with a GDP of $" + str(max\_gdp), "USD.")

**Output**

The country with the highest GDP in the 2020 FY is World with a GDP of $84705400000000.0 USD.

**Summary 2 Code**

# Average GDP per year

import csv

is\_first\_line = True

num\_of\_countries = 0

cumulative\_gdp\_2019 = 0

cumulative\_gdp\_2020 = 0

# average\_gdp\_per\_year is a dictionary with the years in the dataset as the keys and the average gdp from that year as the value.

average\_gdp\_per\_year = {}

cleaned\_gdp = open("gdp-cleaned.csv")

csv\_reader = csv.reader(cleaned\_gdp)

for row in csv\_reader:

if is\_first\_line:

is\_first\_line = False

else:

num\_of\_countries += 1

country = row[0]

gdp\_2019 = float(row[1])

gdp\_2020 = float(row[2])

cumulative\_gdp\_2019 += gdp\_2019

cumulative\_gdp\_2020 += gdp\_2020

average\_gdp\_2019 = cumulative\_gdp\_2019 / num\_of\_countries

average\_gdp\_2020 = cumulative\_gdp\_2020 / num\_of\_countries

average\_gdp\_per\_year[2019] = average\_gdp\_2019

average\_gdp\_per\_year[2020] = average\_gdp\_2020

print(average\_gdp\_per\_year)

**Output**

{2019: 3104246300165.3306, 2020: 2990106351360.441}

**Summary 3 Code**

# Number of countries with above average GDP in 2019

import csv

cleaned\_gdp = open("gdp-cleaned.csv")

csv\_reader = csv.reader(cleaned\_gdp)

is\_first\_line = True

countries\_above\_average = 0

average\_gdp\_2019 = 3104246300165.3306 # from summary 2

for row in csv\_reader:

if is\_first\_line:

is\_first\_line = False

else:

gdp\_2019 = float(row[1])

if gdp\_2019 > average\_gdp\_2019:

countries\_above\_average += 1

print("There are", countries\_above\_average, "countries with a GDP above average in 2019.")

**Output**

There are 29 countries with a GDP above average in 2019.

## 2.4 COVID-19 Tests

### 2.4.1 Exploration and Metadata

**Overview**

The dataset is a combined COVID-19 test and population dataset, available from Kaggle at the URL: <https://www.kaggle.com/okwirjulius/covid19-cases-in-africa>. This dataset is organised by country, containing daily information about the COVID-19 testing performed. This data is available from June 1, 2020. The dataset recorded the test number and the population of various countries from 1st June 2020 to 31st December 2020.

The data is presented in a clear format and contains the following fields for each attribute:

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| Date | Date | date in YY/MM/DD, reported in WHO |
| Country, Other | String | Country, Region, or dependency |
| Total Tests | Integer | cumulative number of tests up till that date |
| Population | Integer | population of Country, Region, or dependency |
| Tests/1M pop | Integer | tests per 1 million of the population |
| 1 Testevery X ppl | Integer | 1 test for every X number of people |

**Data Provenance**

The data was resourced from <https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data>, all the data elements are collected cumulatively. The collection methodologies are from Johns Hopkins University: <https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports> and the Worldometer: <https://www.worldometers.info/> . The collections are updated cumulatively in these two methodologies.

This data is licenced under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence. This means that we have permission to use, transform and distribute the data set as long as the original author is attributed. For this data the original author is the "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University" (<https://github.com/CSSEGISandData/COVID-19>).

**Strengths and Limitations**

The dataset showcases a great strength in comparing the total test number and the population of a country in a straight way. It also calculates the tests per 1 million of the population and the 1 test for every X number of people to clearly analyse how the test number ranges from the whole population in a country.

However, some data is unknown for some countries that couldn’t be found in the WHO or Worldometer website for some reasons, so we can’t extract every country’s valuable information to analyze. It definitely affects the data quality negatively.

### 2.4.2 Data Quality

**Inspection**

The dataset has a pretty high quality from kaggle except some unknown values in some column with none missing values:

**Inspection Code**

is\_first\_line = True

total\_missing\_values = 0

num\_of\_values = 0

for row in open("covid19\_tests.csv"):

if is\_first\_line:

is\_first\_line = False

else:

values = row.split(",")

missing\_values = 0

for value in values:

num\_of\_values += 1

if value == "":

missing\_values += 1

total\_missing\_values += missing\_values

percentage\_missing = (total\_missing\_values/num\_of\_values)\*100

print(total\_missing\_values, "out of", num\_of\_values, "values missing (" + str(round(percentage\_missing, 2)) + "%)")

**Output**

0 out of 434160 values missing (0.0%)

**Cleaning**

The process of data cleaning would be seperated in two ways.

Firstly, searching and deleting some columns with the "NA" values and some error country name from the first row to the last row with python code which are listed in the following section.

Secondly, change the date format to a proper way to integrate which is in "YYYY-MM-DD" format.

**Cleaning Code**

import csv

is\_first\_line = True

csvReader = csv.reader(open("covid19\_tests.csv", "w+",))

contents = list(csvReader)

rows\_delete = []

for row in contents:

rows\_delete.append(row)

if is\_first\_line:

is\_first\_line = False

else:

country = row[1]

if "NA" in row:

rows\_delete.append(row)

if country.isalpha() == False:

rows\_delete.append(row)

for row in rows\_delete:

contents.remove(row)

writer = csv.writer(open("covid19-tests-cleaned", "w+"))

writer.writerows(contents)

### 2.4.3 Preliminary Analysis

The preliminary analysis determines one point to check out the relationship between the total tests and the population density from different countries.

**Summary Code**

dictionary = {}

is\_first\_line = True

for line in open("covid19-tests-cleaned.csv"):

if is\_first\_line:

is\_first\_line = False

else:

values = line.split(",")

date = values[0]

population = values[3]

tests = values[2]

country = values[1]

dic\_key = (date, country, population, tests)

if dic\_key in dictionary:

dictionary[dic\_key].append(tests)

else:

dictionary[dic\_key] = [tests]

lowest = None

for key in dictionary:

maximum = max(dictionary[key])

if lowest == None:

lowest = maximum

info = key

else:

if maximum > lowest:

highest = maximum

info = key

print("Date:", info[0])

print("Country:", info[1])

print("Population:", info[2])

print(" Total Tests number:", info[3],"\n")

**Output**

Date: 2020/6/1

Country: Anguilla

Population: 14993

Total Tests number: 41

Date: 2020/6/1

Country: Armenia

Population: 2962831

Total Tests number: 58668

Date: 2020/6/1

Country: Belgium

Population: 11585802

Total Tests number: 876306

Date: 2020/6/1

Country: Brazil

Population: 212442762

Total Tests number: 930013

Date: 2020/6/1

Country: Faeroe Islands

Population: 48849

Total Tests number: 9677

Date: 2020/6/1

Country: Uganda

Population: 45610068

Total Tests number: 96825

Date: 2020/6/1

Country: Venezuela

Population: 28441746

Total Tests number: 975825

Date: 2020/6/2

Country: Iran

Population: 83906701

Total Tests number: 975936

Date: 2020/6/2

Country: Venezuela

Population: 28441746

Total Tests number: 987902

Date: 2020/6/3

Country: Iran

Population: 83909638

Total Tests number: 997009

Date: 2020/6/6

Country: Brazil

Population: 212459250

Total Tests number: 999836

Date: 2020/11/30

Country: Australia

Population: 25623412

Total Tests number: 9999897

**Conclusion for analysis 1:**

Australia had the maximum total test number with a population density : 25623412.

And Brazil also have a high total test number with a more population density: 212459250

Thus, the total test number of a country is highly connected to its population.

# 3.0 Data Integration

**Overview**

After cleaning each of our datasets, we created an integrated dataset, which incorporates aspects from each of the original dataset. In this integrated dataset, we have recordings of the new cases, new deaths, tests and vaccinations for each country for each date. Additionally, we also include information about the GDP and population of each country, in order to facilitate future analysis. The format of the integrated dataset is as follows:

| **Field Name** | **Type** | **Description** | **Provenance** |
| --- | --- | --- | --- |
| Date | Date | Date on which the data was collected. Represented in standard ISO format as YYYY-MM-DD. | Common to all datasets. Taken in the range of June 1, 2020 - December 30, 2020. |
| Country | String | The country from which the data was recorded. | Common to all datasets. |
| GDP | Integer | The gross domestic product for the current financial year, measured in USD. | World Bank GDP dataset ([2.3](#_b15mgrkm7hnq)). |
| Population | Integer | The population of the country. | Worldometer COVID-19 Tests dataset ([2.4](#_a1p0ci9phjcm)). |
| New Cases | Integer | The new cases of COVID-19 recorded for this date. | World Health Organisation COVID-19 dataset ([2.2](#_aaeky4ie6yyr)). |
| New Deaths | Integer | The new deaths of COVID-19 recorded for this date. | World Health Organisation COVID-19 dataset ([2.2](#_aaeky4ie6yyr)). |
| Tests | Integer | The number of COVID-19 tests administered for this date. | Worldometer COVID-19 Tests dataset ([2.4](#_a1p0ci9phjcm)). |
| Vaccinations | Integer | The number of COVID-19 vaccinations administered for this date. | Our World in Data Vaccination dataset ([2.1](#_i7m8rcmqezyw)). |
| Vaccines Available | String Array | The names of the brand of COVID-19 vaccines available in the country. | Our World in Data Vaccination dataset ([2.1](#_i7m8rcmqezyw)). |

**Integration**

The integration process itself is fairly straightforward, since it utilises the cleaned datasets created in [Section 2.0](#_jmaaw5azl0hj). It begins by opening each cleaned file and storing the relevant data (the data that will be used in the integrated dataset) in memory in several dictionaries. After this is completed, the countries common to each dataset are determined - if one of the fields is missing for a country, then it is not included in the integrated dataset. From here, we simply iterate over each date in the (common) date range, writing a row for each (common) country. Each field in the row is populated from the dictionaries created earlier.

**Challenges Encountered**

During the process of integrating the datasets, our group encountered several issues that required us to carefully consider how best to solve them. In particular, some of the datasets had incompatible formats. The GDP data set used a wide format, while the WHO, Testing, and Vaccination datasets used a long format. In the integrated dataset, this was fixed by creating a single column for GDP and using a denormalized structure where the GDP is repeated for all dates within the same financial year. The same pattern was used for the Population and Vaccines Available columns, for the same reason. The other main issue involved the Country field being different between datasets. We decided to take the country values that were common to all datasets, which would ensure that we had no missing values in any of the rows of our integrated dataset.

**Integration Code**

import csv

def inFinancialYear(date, financialYear):

year = int(date.split("-")[0])

month = int(date.split("-")[1])

if (year == financialYear and month >= 7) or (year == (financialYear + 1) and month < 7):

return True

return False

# Dictionaries of (date -> dictionary of {country -> value}}.

newCasesByCountryByDate = {}

newDeathsByCountryByDate = {}

testsByCountryByDate = {}

vaccinationsByCountryByDate = {}

# Dictionaries of (country -> value}.

gdpByCountry = {}

populationByCountry = {}

vaccinesAvailableByCountry = {}

# GDP DATA

csvReader = csv.reader(open("gdp-cleaned.csv"))

contents = list(csvReader)

isFirstLine = True

for row in contents:

if isFirstLine:

isFirstLine = False

continue

country = row[0]

gdp2019 = row[1]

gdp2020 = row[2]

gdpByCountry[country] = (gdp2019, gdp2020)

# WHO CASES/DEATHS DATA

csvReader = csv.reader(open("WHO-COVID-19-global-data-cleaned.csv"))

contents = list(csvReader)

isFirstLine = True

for row in contents:

if isFirstLine:

isFirstLine = False

continue

date = row[0]

country = row[1]

newCases = row[2]

newDeaths = row[3]

# If the date is not in FY 2019 or FY 2020, do not include it.

if not inFinancialYear(date, 2019) and not inFinancialYear(date, 2020):

continue

if date not in newCasesByCountryByDate:

newCasesByCountryByDate[date] = {}

newCasesByCountryByDate[date][country] = newCases

if date not in newDeathsByCountryByDate:

newDeathsByCountryByDate[date] = {}

newDeathsByCountryByDate[date][country] = newDeaths

# TESTS/POPULATION DATA

csvReader = csv.reader(open("covid19-tests-cleaned.csv"))

contents = list(csvReader)

isFirstLine = True

for row in contents:

if isFirstLine:

isFirstLine = False

continue

date = row[0]

country = row[1]

tests = row[2]

population = row[3]

# If the date is not in FY 2019 or FY 2020, do not include it.

if not inFinancialYear(date, 2019) and not inFinancialYear(date, 2020):

continue

if date not in testsByCountryByDate:

testsByCountryByDate[date] = {}

testsByCountryByDate[date][country] = tests

if country not in populationByCountry:

populationByCountry[country] = population

# VACCINATIONS DATA

csvReader = csv.reader(open("country-vaccinations-cleaned.csv"))

contents = list(csvReader)

isFirstLine = True

for row in contents:

if isFirstLine:

isFirstLine = False

continue

date = row[1]

country = row[0]

vaccinations = row[2]

vaccinesAvailable = row[3]

# If the date is not in FY 2019 or FY 2020, do not include it.

if not inFinancialYear(date, 2019) and not inFinancialYear(date, 2020):

continue

if date not in vaccinationsByCountryByDate:

vaccinationsByCountryByDate[date] = {}

vaccinationsByCountryByDate[date][country] = vaccinations

if country not in vaccinesAvailableByCountry:

vaccinesAvailableByCountry[country] = vaccinesAvailable

writer = csv.writer(open("Group-12-integrated.csv", 'w', newline=''))

headerRow = ["Date", "Country", "GDP", "Population", "New Cases", "New Deaths", "Tests", "Vaccinations", "Vaccines Available"]

writer.writerow(headerRow)

# Test dataset is the most restrictive (date-wise), so use it to determine valid dates.

validDates = sorted(testsByCountryByDate.keys())

# Get the countries common to all datasets.

gdpKeys = set(gdpByCountry.keys())

populationKeys = set(populationByCountry.keys())

vaccinesAvailableKeys = set(vaccinesAvailableByCountry.keys())

newCasesKeys = set(list(newCasesByCountryByDate.values())[0].keys())

newDeathsKeys = set(list(newDeathsByCountryByDate.values())[0].keys())

testsKeys = set(list(testsByCountryByDate.values())[0].keys())

vaccinationsKeys = set(list(vaccinationsByCountryByDate.values())[0].keys())

validCountries = sorted(gdpKeys.intersection(newCasesKeys).intersection(newDeathsKeys).intersection(testsKeys).intersection(populationKeys).intersection(vaccinationsKeys).intersection(vaccinesAvailableKeys))

for date in validDates:

for country in validCountries:

# Calculate the GDP.

if inFinancialYear(date, 2019):

gdp = gdpByCountry[country][0]

else:

gdp = gdpByCountry[country][1]

# Calculate the new cases, new deaths, tests, population and vaccinesAvailable.

newCases = newCasesByCountryByDate[date][country]

newDeaths = newDeathsByCountryByDate[date][country]

tests = testsByCountryByDate[date][country]

population = populationByCountry[country]

vaccinesAvailable = vaccinesAvailableByCountry[country]

# The vaccine rollout began at different times for each country:

# If this date is before the rollout for this country (aka no record),

# then set the value to 0.

if date in vaccinationsByCountryByDate and country in vaccinationsByCountryByDate[date]:

vaccinations = vaccinationsByCountryByDate[date][country]

else:

vaccinations = 0

row = [date, country, gdp, population, newCases, newDeaths, tests, vaccinations, vaccinesAvailable]

writer.writerow(row)