1. **Project Overview**:

This project was initiated to satisfy the requirements of the “Project 2” assignment for the Monash Data Analytical Bootcamp.

The project also provides a much-needed solution for analysing global Covid-19 trends.

These requirements are:

* + 1. Your visualization must include a Python Flask–powered API, HTML/CSS, JavaScript, and at least one database (SQL, MongoDB, SQLite, etc.).
    2. Your project should fall into one of the below four tracks:
       1. A custom “creative” D3.js project (i.e., a nonstandard graph or chart)
       2. A combination of web scraping and Leaflet or Plotly
       3. A dashboard page with multiple charts that update from the same data
       4. A “thick” server that performs multiple manipulations on data in a database prior to visualization (must be approved)
    3. Your project should include at least one JS library that we did not cover.
    4. Your project must be powered by a data set with at least 100 records.
    5. Your project must include some level of user-driven interaction (e.g., menus, dropdowns, textboxes).
    6. Your final visualization should ideally include at least three views.

1. **Team Members**: Megan Greenhill

Hesh Kuruppuge

Jacqueline Xia

Mike Murphy

1. **Project Brief Description**:

The project uses a Python Flask-powered API to access an integrated PostgreSQL database that contains Covid-19 information sourced from the John Hopkins University Covid-19 time series datasets, vaccination information sourced from Our World in Data and population information sourced from Worldometers.

It provides the basis for an ongoing analysis of Covid\_19 and its global impact using user selected visualisations.

1. **Project Rationale**:

There is currently no single source that provides a comprehensive, integrated view of the global Covid-19 pandemic. To get a comprehensive overview you have to use multiple data sources and combine their outputs to get a single view.

This project combines the main sources of global Covid data, global vaccination data and global population data to provide a single, integrated source of information in a relational database and to provide visualisations to retrieve views of the data selected by the user.

The database can be updated as frequently as required and will facilitate analysis that currently requires separate searches of disparate datasets.

1. **Project Datasets**:

The datasets for the project can be found at the following links.

“JHU – Time Series Daily Reports”

<https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv>

“World population data”

<https://www.worldometers.info/world-population/population-by-country/>

“Vaccination rates per country”

<https://ourworldindata.org/covid-vaccinations>?

1. **Database QuickDB Code**

The QuickDB code used to create the data base schema follows.

{\rtf1\ansi\ansicpg1252\cocoartf2580

\cocoatextscaling0\cocoaplatform0{\fonttbl\f0\fswiss\fcharset0 Helvetica;}

{\colortbl;\red255\green255\blue255;}

{\\*\expandedcolortbl;;}

\paperw11900\paperh16840\margl1440\margr1440\vieww11520\viewh8400\viewkind0

\pard\tx566\tx1133\tx1700\tx2267\tx2834\tx3401\tx3968\tx4535\tx5102\tx5669\tx6236\tx6803\pardirnatural\partightenfactor0

\f0\fs24 \cf0 country\_codes\

-\

country\_id VARCHAR(255) PK\

country\_name VARCHAR(255)\

continent\_name VARCHAR(255)\

\

covid\_cases\

-\

country\_id VARCHAR(255) FK - country\_codes.country\_id\

date VARCHAR(255)\

confirmed INT\

deaths INT\

recovered INT\

active INT\

new\_cases INT\

new\_deaths INT\

new\_recovered INT\

\

population\

-\

country\_id VARCHAR(255) FK - country\_codes.country\_id\

population INT\

\

vaccinations\

-\

country\_id VARCHAR(255) FK - country\_codes.country\_id\

date VARCHAR(255)\

vaccinated\_per\_hundred INT\

fully\_vaccinated\_per\_hundred INT\

not\_fully\_vaccinated\_per\_hundred INT\

boosted\_per\_hundred INT}

1. **Database Schema – Entity Relationship Diagram**

Text

Description automatically generated

1. **Database Description**

The key to the data base was to use the International Standards Organisation (iso\_code: ISO 3166-1 alpha-3 – three-letter country code) henceforth referred to as “iso-code”, to create relationships between the tables.

The “country-codes” table contains the “iso-code” and matching “country-name” for all countries covered by the “iso-code” and was generated during the Extraction phase of the project.

The “covid-cases” table contains the basic cleansed data from the JHU Data Sets which form the basis of the global view of Covid-19 cases.

The “vaccinations” table contains vaccination status from the Our World in data Vaccination data set.

1. **Database Meta Data**

**“country” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* country-name: the name of the country in the ISO data set

**“covid-cases” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* date: date of the observation
* confirmed: the total number of cumulative confirmed Covid-19 cases regardless of the variant
* deaths: the total number of cumulative deaths attributed to Covid-19 regardless of the variant
* recovered: the total number of cumulative recovered Covid-19 cases
* active: the total number of cumulative active Covid-19 cases
* new\_cases: the total number of incremental new Covid-19 cases
* new\_deaths: the total number of incremental new Covid-19 deaths
* new\_ recovered:

the total number of incremental new recovered Covid-19 cases

**“population” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* population: the population of the country at 31/12/2020

**“vaccinations” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* date: date of the observation
* vaccinated\_per\_hundred:

total number of people who received at least one vaccine dose. If a person receives the first dose of a 2-dose vaccine, this metric goes up by 1. If they receive the second dose, the metric stays the same i.e., 1.

* fully\_vaccinated\_per\_hundred:

people vaccinated per 100 people in the total population of the country. If a person receives the first dose of a 2-dose vaccine, this metric stays the same. If they receive the second dose, the metric goes up by 1.

* not\_fully\_vaccinated\_per\_hundred:

 people not vaccinated per 100 people in the total population of the country

* boosted\_per\_hundred:

people who have received their booster dose per 100 people in the total population of the country

1. **Data Extract:**

The Extract phase of the assignment uses urls / wget downloads in place of API calls are they are not available for the datasets needed. The three JHU time series data sets are retrieved using this method.

The Vaccination and Population data sets are downloaded from their respective sites using the pd.read\_csv function.

1. **Data Transform:**

The data Transform steps are as follows:

1. Save DFs to CSVs to do exploratory data analysis.
2. Conduct exploratory data analysis.
3. Use melt() to unpivot DataFrames from current wide format 265 rows × 749 columns into long format 208600 rows × 6 columns.
4. Remove recovered data for Canada due to mismatch issue. Canada recovered data is counted for the whole Country instead of by Province/State which is how Canada and the rest of the world count data for "Confirmed Cases" and "Deaths".

We could apportion recovered data for the country across the Province/States in the same ratio as confirmed cases time permitting

1. Merge the three JHU dataframes, Confirmed Cases, Deaths, Recovered Cases.
   1. merge confirmed\_df\_long and deaths\_df\_long into full\_table
   2. merge full\_table and recovered\_df\_long
2. Check Canada data in "full\_table" - "recovered" should be 0 and check of CSV file confirms that it is.
3. Convert date from string to datetime.
4. Detect missing values NaN.
5. Replace 'recovered' NaNs with zero.
6. Three cruise ships need to be treated differently to the rest of the cases.So extract and remove data for these ships.
7. Calculate active cases = confirmed cases - deaths – recovered cases.
8. Aggregate data into Country/Region and group by Date and Country/Region.
9. Calculate daily New cases, New deaths and New recovered by deducting the corresponding accumulative data on the previous day
10. Use pd.merge to group the final data frame on Country/Region / Date.
11. Fix the new data types as integer.
12. The final data frame is sorted by Date and Country/Region ascending where: -

Confirmed Cases, Deaths, Recovered and Active are cumulative data for the entire period, and,

New cases, New deaths and New Recovered are daily incremental data.

1. Convert data frame to a csv file for backup.
2. Select Australia to check that data is correct. Validate the final data frame against the JHU Dashboard for 06/02/2022.

Both showed Confirmed Cases = 2,704,275 and Deaths = 4,154 for Australia.

1. Read the Vaccination dataset - csv file into a data frame.
2. Derive the “people\_not\_vaccinated” from the “people\_fully\_vaccinated”.
3. Detect missing values NaN
4. Replace NaNs with zero
5. Data cleansing replace ”United States” with “US” to standardise data.
6. Save cleansed vaccination data to a CSV for backup.
7. Read the Population data set - csv file into a data frame.
8. Detect missing values NaN
9. Replace NaNs with zero
10. Save cleansed Population data to a CSV for backup.
11. Copy OWID Vaccination data frame, as we want to use OWID country codes.
12. Add Africas to match population data frame.
13. Edit “full\_grouped” covid case data frame to include country ID.
14. Change structure of data frames to match structure of tables created in the database.
15. Set index of country codes data frame and remove null index row.
16. Covid Cases table - copy only the columns needed into a new Data Frame.
17. Rename columns to fit the tables created in the database.
18. Vaccinations table - copy only the columns needed into a new Data Frame.
19. Rename columns to fit the tables created in the database.
20. Create PostgreSQL database connection.
21. Confirm database tables.
22. Load data frames to database tables
23. **Data Analysis Using Pandas Profiling:**

We used Pandas Profiling to analyse the Confirmed Cases, Vaccinations and the World Population data frames produced by the “create-covid-db” notebook to ensure their data integrity. The results are as follows.

1. **EDA – Confirmed Cases**

The exploratory data analysis for Confirmed Cases ran successfully and the Output Report is located at: <file:///C:/Users/mikem/Documents/MDAB_homework/13-project-2/output_conf_covid_cases.html>

1. **EDA – Vaccinations**

The exploratory data analysis for Vaccinations ran successfully and the Output Report is located at:

<file:///C:/Users/mikem/Documents/MDAB_homework/13-project-2/output_conf_vaccinations.html>

1. **EDA – World Population**

The exploratory data analysis for World Population ran successfully and the Output Report is located at:

<file:///C:/Users/mikem/Documents/MDAB_homework/13-project-2/output_conf_world_population.html>