Creating Google Cloud Platform (GCP) ecosystem for research on Covid-19 dataset for WHO

Keywords: GCP, Big Query, Covid-19, Data Visualization, GCP Log Analytics

Research Lab is acknowledged the contributors from Kyndryl Solutions Private Limited for developing, design and implementation in GCP ecosystem.

Authors :

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Project Contributors** | **Institution** | **Email ID** |
| Ramamurthy Valavandan | GCP Solution Architect | Kyndryl Solutions Private Limited | Ramamurthy.Valavandan@kyndryl.com |
| Musheer Ahmed N | Project Manager | Kyndryl Solutions Private Limited | Musheer.Ahmed.N@kyndryl.com |
| Ubaiyadulla Sherif | GCP Engineer | Kyndryl Solutions Private Limited | Ubaiyadulla.Sherif@kyndryl.com |
| Pallav Kohli | GCP Developer | Kyndryl Solutions Private Limited | Pallav.Kohli@kyndryl.com |
| Vikram Sharma | GCP Architect | Kyndryl Solutions Private Limited | Vikram.Sharma1@kyndryl.com |
| Pragathi S | GCP Engineer | Kyndryl Solutions Private Limited | pragathi.s@kyndryl.com |
| Vijay R | Cloud Engineer | Kyndryl Solutions Private Limited | vijay.r1@kyndryl.com |
| Surasa Mukherjee | GCP Architect | Kyndryl Solutions Private Limited | surasa.mukherjee@kyndryl.com |
| Kumaraswamy Reddy | GCP Engineer | Kyndryl Solutions Private Limited | Kumaraswamy.Yanamala@kyndryl.com |
| Prasanth Parayatham | GCP Architect | Kyndryl Solutions Private Limited | Prasanth.Parayatham@kyndryl.com |
| Nitin Ambekar | Cloud Engineer | Kyndryl Solutions Private Limited | nitin.ambekar@kyndryl.com |
| Dinesh Sai Teja Neeli | CGP Engineer | Kyndryl Solutions Private Limited | dinesh.sai.teja.neeli@kyndryl.com |
| Santosh Baran | GCP Guild Manager | Kyndryl Solutions Private Limited | Santosh.Baran@kyndryl.com |
| Vijender Singh | Cloud Engineer | Kyndryl Solutions Private Limited | vijender.singh@kyndryl.com |
| Saurabh Uniyal | GCP Database Engineer | Kyndryl Solutions Private Limited | Saurabh.Uniyal@kyndryl.com |
| Praveen B | Cloud Engineer | Kyndryl Solutions Private Limited | praveen.b@kyndryl.com |

Kyndryl Solutions Private Limited, Google Cloud Platform (GCP) Guild Project, 6th Floor, D-1, Manyata Tech Park, Bangalore 560024, Karnataka, India

Motivation:

Create the Google Cloud Platform (GCP) ecosystem for WHO dataset in vaccination for research and analytics for data visualization. In the process of access to the WHO dataset available in the WHO portal helps the researchers and stake holders and vision on the data visualization in the design and development. Also helps the researchers to give insight on the vaccination data and add value to be beneficiary.

Beneficiary: World Health Organization (WHO) research operation, Nature Labs (United Nations Body)

Project is designed and developed for the motivation of opensource in healthcare. The project is tax exception and noncommercial research program.

Project team: Google Cloud Platform Guild Team, Kyndryl Solutions Private Limited.

Business requirement: Identification of sources of data and audit log of GCP resources and Services.

Project outcome: Research papers and filing of patent for GCP Guild team.

Score of work: Create the ecosystem in Google Cloud Platform for public community, healthcare research, medical practitioners, government, private body to access the Google data for the scope for decision making in healthcare domain.

Purpose of the research: By publishing the sources of data ingestion from various sources paper and getting the insight of the Covid-19 data of WHO and development of Line of Treatment and collect the data clenching and provide complete end-to-end ecosystem in Google Cloud Platform (GCP).

Task: To create the ecosystem in Google Cloud Platform and Dataset with details of audit log of GCP.

Source: WHO Coronavirus (COVID-19)data

Type of the data: Data is available in the comma-separated values (CSV) files

Research is carried out by the motivation of providing the secure data ingestion, ecosystem, development of line of treatment, decision making in healthcare domain as per the WHO standards in healthcare.

Introduction: The WHO Health Emergency Dashboard is not a comprehensive representation of all the events and emergencies that WHO is aware of and responding to. The events displayed are a subset of those reported through official channels as mandated by the International Health Regulations (IHR 2005). The content of the WHO Health Emergency Dashboard is for general information only.

The size of the charts corresponds to the number of events in each country or territory; the size does not indicate the severity or risk associated with the event(s).

#### WHO Vaccination data

| **Field name** | **Type** | **Description** |
| --- | --- | --- |
| COUNTRY | String | Country, territory, area |
| ISO3 | String | ISO Alpha-3 country code |
| WHO\_REGION | String | WHO regional offices: WHO Member States are grouped into six WHO regions: 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), and Regional Office for the Western Pacific (WPRO). |
| DATA\_SOURCE | String | Indicates data source: - REPORTING: Data reported by Member States, or sourced from official reports - OWID: Data sourced from Our World in Data: https://ourworldindata.org/covid-vaccinations |
| DATE\_UPDATED | Date | Date of last update |
| TOTAL\_VACCINATIONS | Integer | Cumulative total vaccine doses administered |
| PERSONS\_VACCINATED\_1PLUS\_DOSE | Decimal | Cumulative number of persons vaccinated with at least one dose |
| TOTAL\_VACCINATIONS\_PER100 | Integer | Cumulative total vaccine doses administered per 100 population |
| PERSONS\_VACCINATED\_1PLUS\_DOSE\_PER100 | Decimal | Cumulative persons vaccinated with at least one dose per 100 population |
| PERSONS\_FULLY\_VACCINATED | Integer | Cumulative number of persons fully vaccinated |
| PERSONS\_FULLY\_VACCINATED\_PER100 | Decimal | Cumulative number of persons fully vaccinated per 100 population |
| VACCINES\_USED | String | Combined short name of vaccine: “Company - Product name” |
| FIRST\_VACCINE\_DATE | Date | Date of first vaccinations. Equivalent to start/launch date of the first vaccine administered in a country. |
| NUMBER\_VACCINES\_TYPES\_USED | Integer | Number of vaccine types used per country, territory, area |
| PERSONS\_BOOSTER\_ADD\_DOSE | Integer | Persons received booster or additional dose |
| PERSONS\_BOOSTER\_ADD\_DOSE\_PER100 | Decimal | Persons received booster or additional dose per 100 population |

Python Program for getting vaccination data

|  |
| --- |
| """  Client : WHO - Nature Labs  Project Scope : GCP as Infrastructure Modernization  Functional scope : Function to create 'Use Cases' in Google Cloud Platform  GCP Projects Used : Google Storage, Compute Engine  Development : Client to storage the files and provide the access on-demand in GCP  Generate the Google Storage and Generate the compute engine for performance  Written by Kyndryl for gcp Data store location in Nature Labs Project  Author : ramamurthy.valavandan@kyndryl.com  gcloud components  """  import requests  import re  import shutil  import os  import sys  who\_data\_url = 'https://covid19.who.int/who-data/vaccination-data.csv'  whodata=re.sub(r'^.+/([^/]+)$', r'\1', who\_data\_url)  workingdirctory="ds3"  customerdirctory="covid19"  basepath = "C:\\nature-labs\\who"  gcloudcodepaths = ("{}{}{}{}{}".format(basepath,"\\",customerdirctory,"\\",workingdirctory))  fullyqualifiedwhodata = ("{}{}{}".format(gcloudcodepaths,"\\",whodata))  def prt(p):  width = len(p) + 4  print('┏' + "━"\*width + "┓")  print('┃' + p.center(width) + '┃')  print('┗' + "━"\*width + "┛")  #remove file if exists  def remove\_if\_exists(removefile):  try:  if os.path.exists(removefile):  os.remove(removefile)  #print ("File removed successfully", removefile)  pi="\'File removed successfully \' :"  p = ("{}{}".format(pi,removefile))  prt(p)  except:  print("Error while deleting file ", removefile)  #remove previous log file  removefile = fullyqualifiedwhodata  remove\_if\_exists(removefile)  pi="\'Downloading WHO Vaccination data \' :"  p = ("{}{}".format(pi,who\_data\_url))  prt(p)  def downloading(download\_url,local\_file\_data):  file\_stream = requests.get(download\_url, stream=True)  with open(local\_file\_data, 'wb') as local\_file:  for data in file\_stream:  local\_file.write(data)  download\_url=who\_data\_url  local\_file\_data=fullyqualifiedwhodata  downloading(download\_url,local\_file\_data)  pi="\'Download is completed : \' :"  p = ("{}{}".format(pi,fullyqualifiedwhodata))  prt(p) |

The above python code generates the vaccination data in CSV format in c:\nature-labs\who\who-ds2 directory

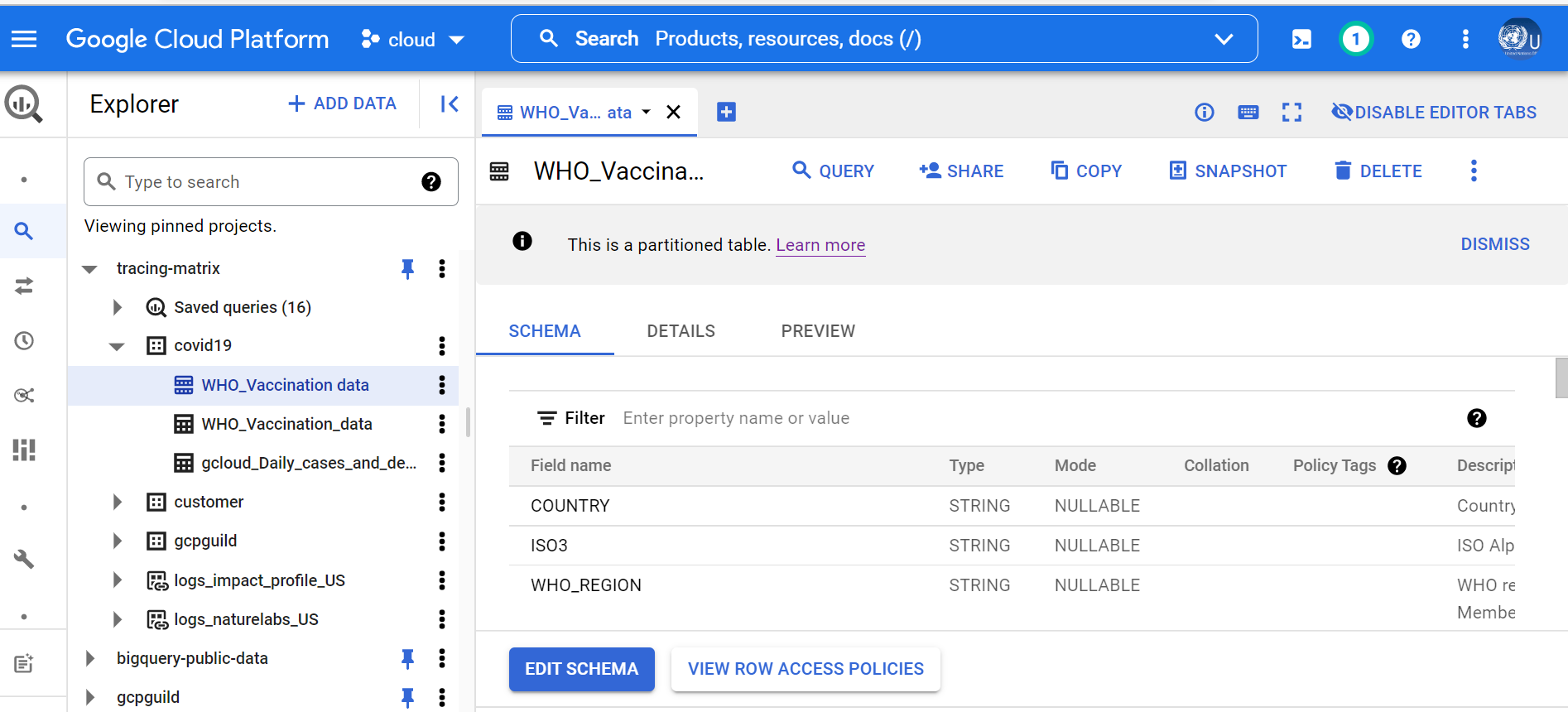
The below code uses the projectID="tracing-matrix" and dataset="covid19" in GCP – Nature Labs.

|  |
| --- |
| """  Client : WHO - Nature Labs  Project Scope : GCP as Infrastructure Modernization  Functional scope : Function to create 'Use Cases' in Google Cloud Platform  GCP Projects Used : Google Storage, Compute Engine  Development : Client to storage the files and provide the access on-demand in GCP  Generate the Google Storage and Generate the compute engine for performance  Written by Kyndryl for gcp Data store location in Nature Labs Project  Author : ramamurthy.valavandan@kyndryl.com  gcloud components  """  ipfile="vaccination-data.csv"  #ifl="Latest reported counts of cases and deaths"  currentdirds="ds3"  coviddir="covid19"  basepath = "C:\\nature-labs\\who"  gcli="WHO"  #C:\nature-labs\who\covid19\ds3  projectID="tracing-matrix"  dataset="covid19"  URI="https://drive.google.com/file/d/132PDmI2o9gParYa4F23o\_IqdR8Ncxo0J/view?usp=sharing"  ifc="Vaccination data"  sl=27  ls=len(ifc)  if(ls <= sl ):  ifl=ifc  else:  ifl=(ifc[0:sl])  """  NO CHANGE SHOULD BE DONE AFTERWARDS ...  """  gcloudcodepaths = ("{}{}{}{}{}".format(basepath,"\\",coviddir,"\\",currentdirds))  chkwho = ("{}{}{}".format(gcloudcodepaths,"\\",ipfile))  import re  import glob  from tkinter import W  import pandas as pd  from pandas import ExcelWriter  from pandas import ExcelFile  from os.path import expanduser as ospath  from pathlib import Path  import logging  import socket  from inspect import getsourcefile  import chardet  import pandas as pd  from datetime import datetime  import shutil  import xlrd  import runpy  import os  import sys  logf ="gcplog.txt"  logfi = ("{}{}".format("\\",logf))  logfile = (gcloudcodepaths + logfi)  logging.basicConfig(  filename = logfile,  level = logging.INFO,  format = '%(levelname)s:%(asctime)s:%(message)s')  logging.info('Compute Engine Directory: %r', {gcloudcodepaths})  fileinfo=(os.path.split(sys.argv[0])[1])  hostname=(socket.gethostbyaddr(socket.gethostname())[0])  datestamp = datetime.now().date()  logging.info('------Start of Google Log Analytics Projects ---------')  logging.info('Host Name %r, Compute Engine = %r', hostname, fileinfo)  path = Path(chkwho)  def prt(p):  width = len(p) + 4  print('┏' + "━"\*width + "┓")  print('┃' + p.center(width) + '┃')  print('┗' + "━"\*width + "┛")  if path.is\_file():  pi="\'Excel file is created \' :"  p = ("{} {}".format(pi,chkwho))  prt(p)    else:  pi="\'excelfilefordtye is missing !\' :"  p = ("{} {}".format(pi,chkwho))  prt(p)  logging.error('Could not find xls file : %r', {p})  exit(1)  targetdir = ("{}{}{}".format(basepath,"\\",currentdirds))  sno=("{}\_{}".format(gcli,ifc))  dc=sno  dc = re.sub('[^A-Za-z0-9]+', ' ', dc)  dc = dc.strip()  dc = dc.rstrip()  dc = dc.lstrip()  dc = re.sub("\s", "\_", dc)  N="\\"  csvout = ("{}{}.{}".format(N,dc,"csv"))  pi = "\'Before Renaming \' :"  p = ("{}\t{}".format(pi,chkwho))  prt(p)  csvfileforuploadcsv = (gcloudcodepaths + csvout)  pi="\'After Renaming \' : "  p = ("{}\t{}".format(pi,csvfileforuploadcsv))  prt(p)  shutil.copy(chkwho, csvfileforuploadcsv)  sno=("{}\_{}".format(gcli,ifl))  fno=("{}\_{}".format(gcli,ifc))  filetylst=['sql','csv','xlsx']  dc=fno  dc = re.sub('[^A-Za-z0-9]+', ' ', dc)  dc = dc.strip()  dc = dc.rstrip()  dc = dc.lstrip()  dc = re.sub("\s", "\_", dc)  ext\_table\_name=dc  N="\\"  for fl in (filetylst):  thr=("{}.{}".format(dc,fl))  if(fl == 'sql'):  bqf = ("{}{}{}".format(N,'BQ\_',thr))  if(fl== 'csv'):  incsv = ("{}{}".format(N,thr))  csvout = ("{}{}{}{}".format(N,'Upload\_','GCP\_',thr))  else:  outxls = ("{}{}".format(N,thr))  infc = (gcloudcodepaths + incsv)  conxls = (gcloudcodepaths + outxls)  csvfileforupload = (gcloudcodepaths + csvout)  bqfile = (gcloudcodepaths + bqf)  excelfilefordtye=csvfileforupload  with open(infc, 'rb') as f:  enc = chardet.detect(f.read())    dfc = pd.read\_csv(infc, encoding = enc['encoding'])  dfc.to\_excel(conxls, sheet\_name=sno, index=False)  excelfilefordtye = conxls  xl = pd.ExcelFile(excelfilefordtye)  sheetlst=xl.sheet\_names  for sn in (sheetlst):  sheetname=sn  logging.info('Sheet Name : %r', sheetname)  with open(excelfilefordtye, "rb") as f:  df\_input\_file = pd.read\_excel(f, sheet\_name=sheetname, header=0, index\_col=None)  colname=df\_input\_file.columns  datatypes=dict(df\_input\_file.dtypes)  row\_count=df\_input\_file.count()[0]  logging.info('No of rows in Input File %r, Row Count %r', excelfilefordtye, row\_count)  logging.info('\nGenerating the Project account in GCP:\n')  logging.info('Google Log Analytics Projects Generated File %r', conxls)  df\_input\_file.head(row\_count).to\_csv(csvfileforupload, encoding='utf-8', header=False, index=False)  def switch(check\_data\_type):  dict={  'object': 'STRING',  'int64' : 'INT64',  'float64': 'FLOAT64',  'DATE' : 'DATE'  }  return dict.get(check\_data\_type, 'Unable to find Data Type')  datearray=['date', 'DATE', 'Date']  fldnames=[]  for fld in colname:  for cdatesrt in (datearray):  check\_date\_return = fld.find(cdatesrt)  check\_date\_lu=cdatesrt  if(check\_date\_return != -1):  check\_data\_type='DATE'  break    else:  dtdef=df\_input\_file[fld].dtypes  check\_data\_type = str(dtdef)  logging.info('Field Name %r, Check Data Type %r, Check DATE Return code %r', {fld}, {check\_data\_type}, {check\_date\_return})      flddty=switch(check\_data\_type)  pi="\'Check Data Type of Field is Date:\' : "  p = ("{}:{}:{}".format(pi,fld,flddty))  dc=fld    dc = re.sub('[^A-Za-z0-9]+', ' ', dc)  dc = dc.strip()  dc = dc.rstrip()  dc = dc.lstrip()  dc = re.sub("\s", "\_", dc)  dc = re.sub(r"[^\w\s]", '', dc)  dc = re.sub(r"\s+", '\_', dc)  ddc=dc  logging.info('Field Name : %r , Data Type :%r ', dc, flddty)  logging.info('Fld Name : %r, Original DTy %r: Converted DTy is : %r', ddc, check\_data\_type, flddty)  tblsting=("{} {}".format(dc, flddty))  fldnames.append(tblsting)    logging.info('Elements in Table Field and Datatype %r', fldnames)  L=[]  lc=1  ll=len(fldnames)  logging.info('Number of Elements in Tbl Fld and Dty List or Array %r', ll)  for fldy in (fldnames):  logging.info('Field and DTy :%r ', fldy)  fldy=("{}\t{}".format("\t",fldy))  L.append(fldy)  if (lc == ll):  N="\n"  else:  N=",\n"  lc += 1  L.append(N)  cene = open(bqfile, 'w')  tbe=ext\_table\_name  tbe= re.sub('[^A-Za-z0-9]+', ' ', tbe)  tbe = tbe.strip()  tbe = tbe.rstrip()  tbe = tbe.lstrip()  tbe = re.sub("\s", "\_", tbe)  tbe = re.sub(r"[^\w\s]", '', tbe)  tbe = re.sub(r"\s+", '\_', tbe)    ts=("{} {} {}".format("--Generated schema for table:",tbe,"--"))  fulltblname=("{}.{}.{}".format(projectID,dataset,tbe))  line1=("{} {}{}{}".format("CREATE EXTERNAL TABLE","`",fulltblname,"`\n"))  line2="(\n"  line3="\n)"  s = """  OPTIONS(  skip\_leading\_rows=0,  format="CSV",  """  uris=("{}{}{}{}".format("uris=[","\"",URI,"\"]"))  line4="\n);"  cene.write(line1)  cene.write(line2)  cene.writelines(L)  cene.write(line3)  cene.write(s)  cene.write(uris)  cene.write(line4)  cene.close()  logformatfile = ("{}{}\_{}\_{}".format("\\",hostname,datestamp,logf))  logdfilenew = (gcloudcodepaths + logformatfile)  shutil.copy(logfile, logdfilenew)  postscript="cleanfiles.py"  cfls = ("{}{}".format("\\",postscript))  cleanfile = (gcloudcodepaths + cfls)  clean = open(cleanfile, 'w')  def cleanfl (rmv, removefile):    ldc = str(removefile)  slfs = (ldc.split('\\'))  leba=len(slfs)-1  for rf in range(0,len(slfs)):    if(rf == 0):  sla=("{}{}".format(rmv," = \""))    else:  sla=("{}{}".format("\\","\\"))  arf=("{}{}".format(sla,slfs[rf]))  larys.append(arf)  if(rf == leba):  dq=("{}{}".format("\"","\n"))  larys.append(dq)  clean.writelines(larys)  removefile=logfile  larys=[]  cleanfl('logfile\_rm',removefile)  larys=[]  cleanfl('excel\_rm',excelfilefordtye)  s = """  import shutil  import os  import sys  #remove file if exists  def remove\_if\_exists(removefile):  try:  if os.path.exists(removefile):  os.remove(removefile)  print ("File removed successfully", removefile)  except:  print("Error while deleting file ", removefile)  #remove previous log file  removefile = logfile\_rm  remove\_if\_exists(removefile)  removefile = excel\_rm  remove\_if\_exists(removefile)  """  clean.write(s)  clean.close  pi="\'Create 'Table' GCP Big Query \' : "  p = ("{} {}".format(pi,bqfile))  prt(p)  pi="Execute 'python' for cleaning file(s) : "  p = ("{} {}".format(pi,cleanfile))  prt(p)  pi="python "  p = ("{} {}".format(pi,cleanfile))  prt(p) |

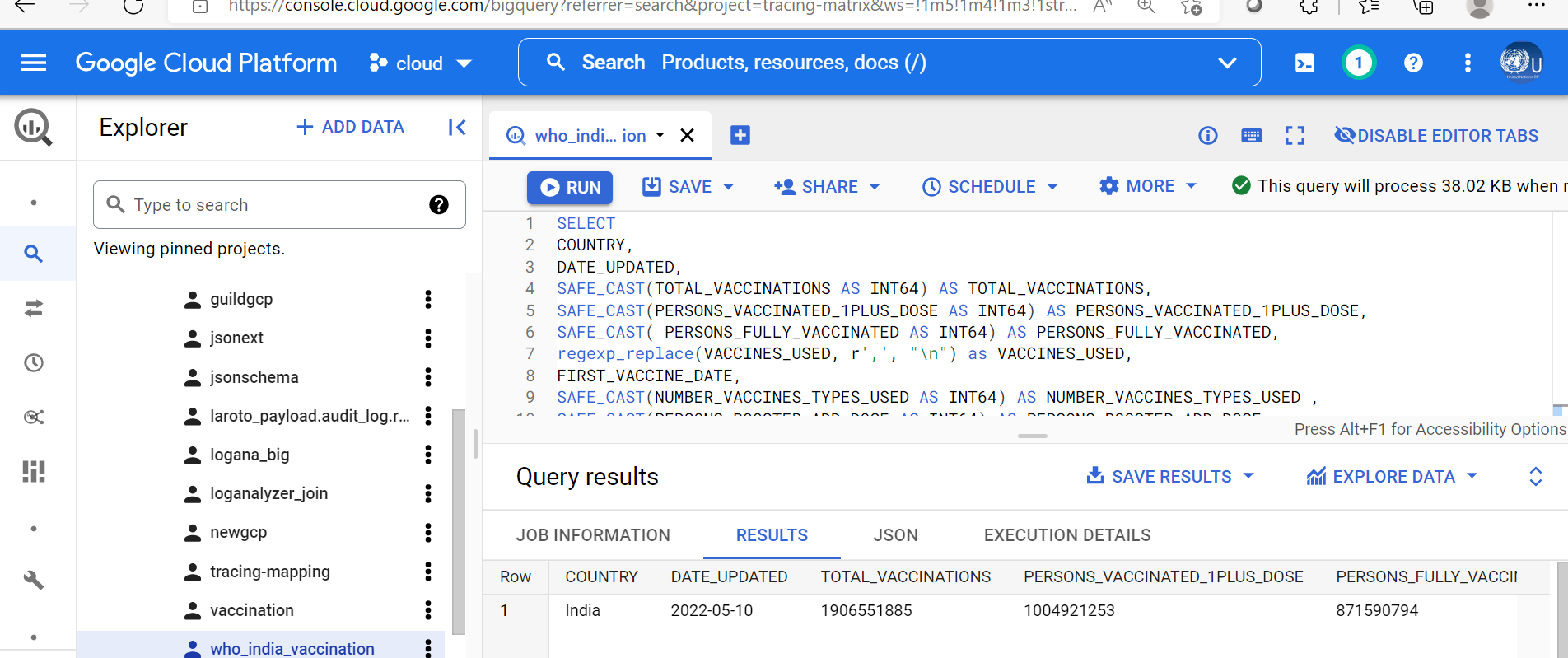
The Big Query SQL is created for external table dataset in the ‘tracking-matrx’

|  |
| --- |
| CREATE EXTERNAL TABLE `tracing-matrix.covid19.WHO\_Vaccination\_data`  (  COUNTRY STRING,  ISO3 STRING,  WHO\_REGION STRING,  DATA\_SOURCE STRING,  DATE\_UPDATED DATE,  TOTAL\_VACCINATIONS FLOAT64,  PERSONS\_VACCINATED\_1PLUS\_DOSE FLOAT64,  TOTAL\_VACCINATIONS\_PER100 FLOAT64,  PERSONS\_VACCINATED\_1PLUS\_DOSE\_PER100 FLOAT64,  PERSONS\_FULLY\_VACCINATED FLOAT64,  PERSONS\_FULLY\_VACCINATED\_PER100 FLOAT64,  VACCINES\_USED STRING,  FIRST\_VACCINE\_DATE DATE,  NUMBER\_VACCINES\_TYPES\_USED FLOAT64,  PERSONS\_BOOSTER\_ADD\_DOSE FLOAT64,  PERSONS\_BOOSTER\_ADD\_DOSE\_PER100 FLOAT64  )  OPTIONS(  skip\_leading\_rows=0,  format="CSV",  uris=["https://drive.google.com/file/d/132PDmI2o9gParYa4F23o\_IqdR8Ncxo0J/view?usp=sharing"]  ); |

Results of the GCP ecosystem

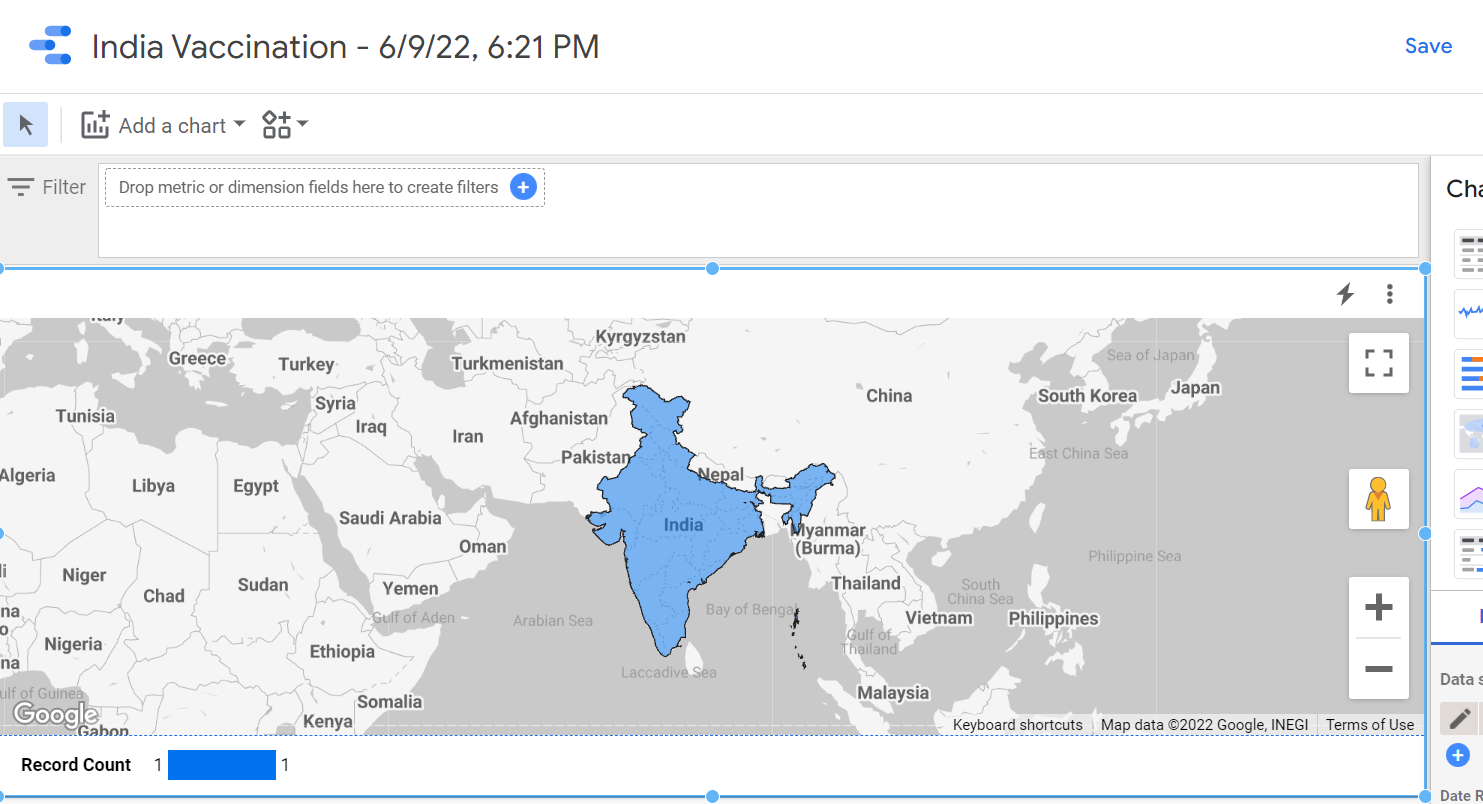


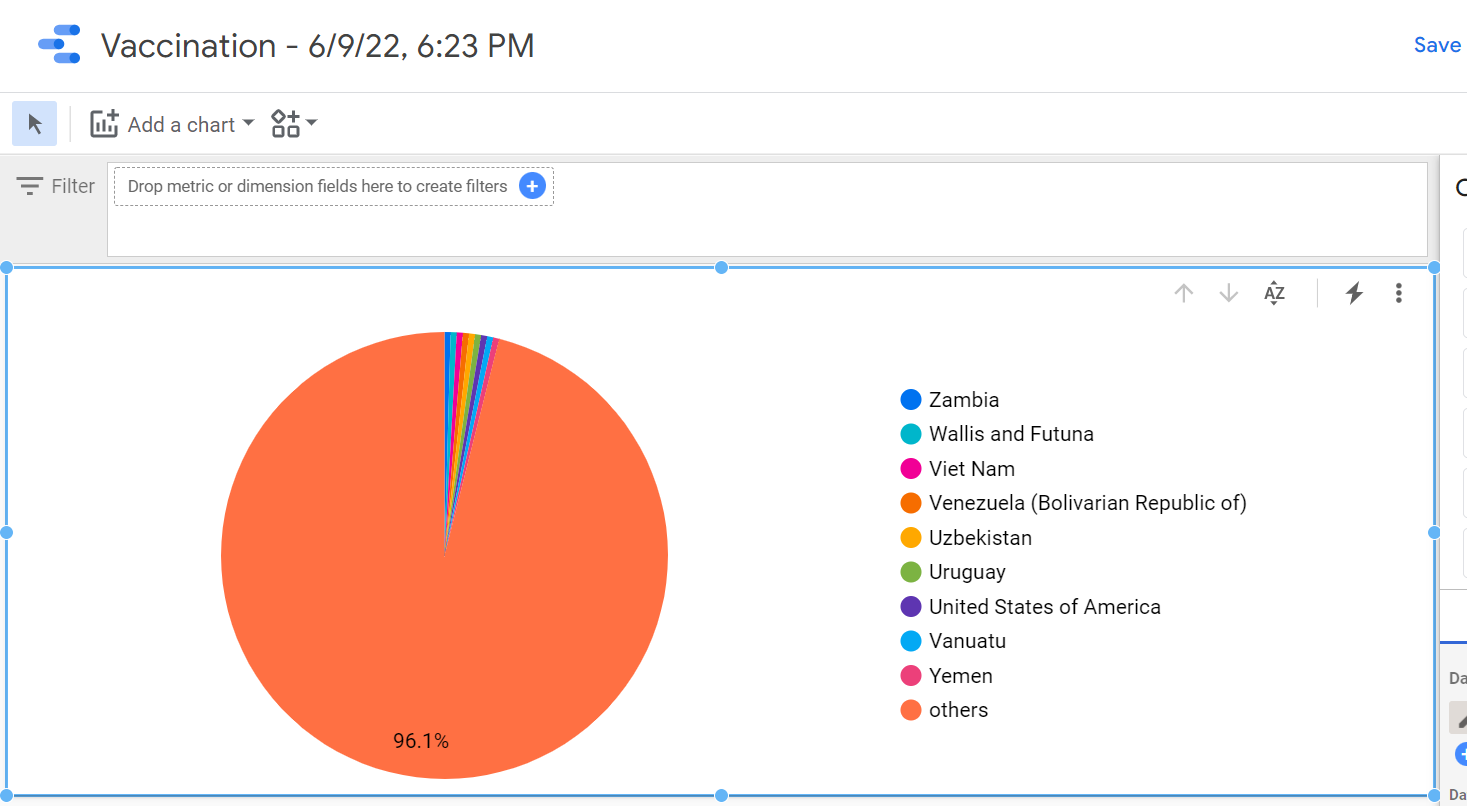
|  |
| --- |
| SELECT  COUNTRY,  DATE\_UPDATED,  SAFE\_CAST(TOTAL\_VACCINATIONS AS INT64) AS TOTAL\_VACCINATIONS,  SAFE\_CAST(PERSONS\_VACCINATED\_1PLUS\_DOSE AS INT64) AS PERSONS\_VACCINATED\_1PLUS\_DOSE,  SAFE\_CAST( PERSONS\_FULLY\_VACCINATED AS INT64) AS PERSONS\_FULLY\_VACCINATED,  regexp\_replace(VACCINES\_USED, r',', "\n") as VACCINES\_USED,  FIRST\_VACCINE\_DATE,  SAFE\_CAST(NUMBER\_VACCINES\_TYPES\_USED AS INT64) AS NUMBER\_VACCINES\_TYPES\_USED ,  SAFE\_CAST(PERSONS\_BOOSTER\_ADD\_DOSE AS INT64) AS PERSONS\_BOOSTER\_ADD\_DOSE  FROM    `tracing-matrix.covid19.WHO\_Vaccination data`  WHERE    DATE(\_PARTITIONTIME) = "2022-05-12"  AND  COUNTRY='India'  GROUP BY 1,2,3,4,5,6,7,8,9  ORDER BY 1 DESC |

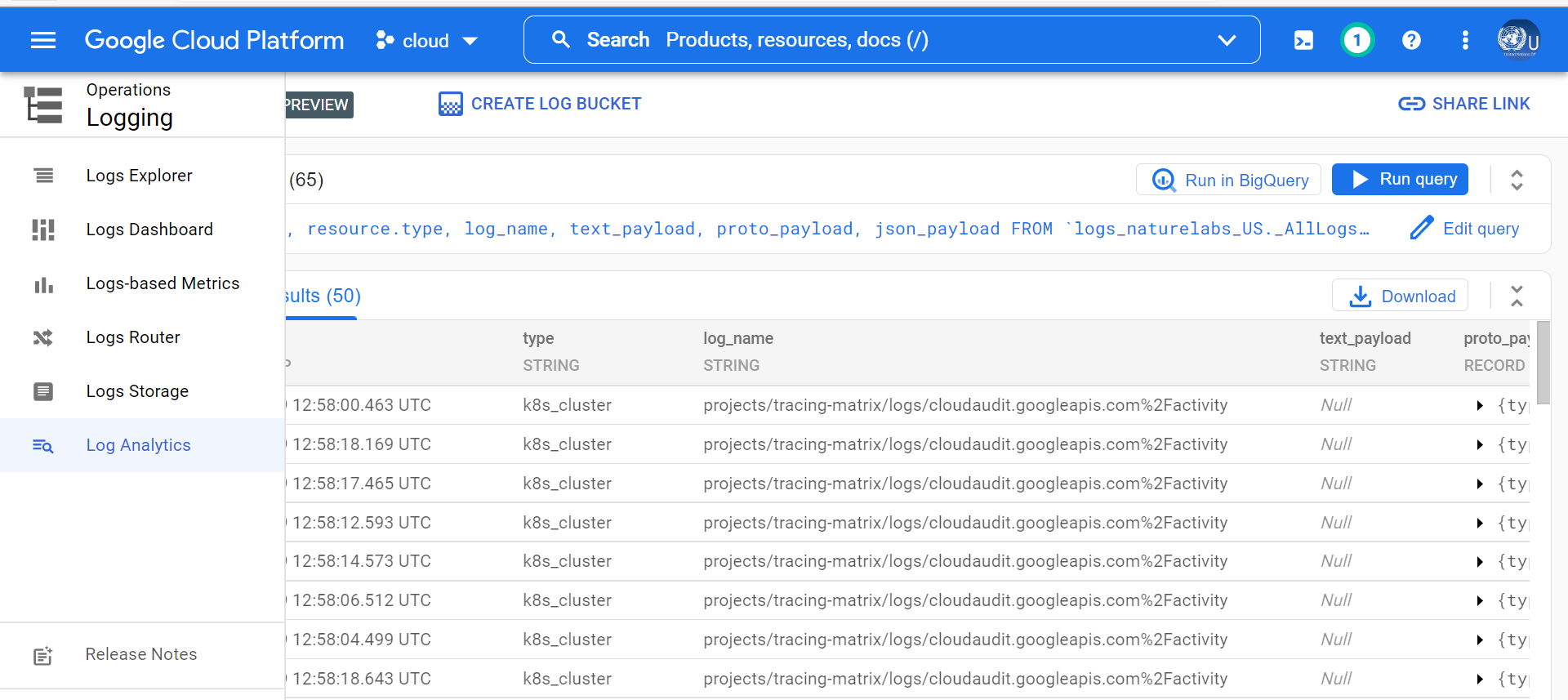


To improve the bi query to improve the performance of the GCP, below \_partitiontime for pseudo partitioning

|  |
| --- |
| SELECT    \*  FROM    `tracing-matrix.covid19.WHO\_Vaccination data`  WHERE    DATE(\_PARTITIONTIME) = "2022-05-12"    AND COUNTRY = 'India' |







|  |
| --- |
| SELECT    timestamp, resource.type, log\_name, text\_payload, proto\_payload, json\_payload  FROM    `logs\_naturelabs\_US.\_AllLogs`  WHERE    timestamp > TIMESTAMP\_SUB(CURRENT\_TIMESTAMP(), INTERVAL 100 DAY)  LIMIT 50 |

Above log analytics helps to identify the location of the data ingestion.

- Ethical Approval and Consent to participate (should this be indicated as it is a health care based on Blockchain and Privacy Computing) : Nature Labs is United Nations research wing in Covid-19 has involved Kyndryl Solutions Private Limited for the dataset creation, python, Big Query and Compute Engine.

- Consent for publication (should this be indicated as it is a health care based on Blockchain and Privacy Computing)

Google Cloud Platform (GCP), big query is allowed by the Kyndryl Solutions Private Limited in WHO dataset in the cloud platform. All the authors are pleased to support to the publishers and any further communication from the readers and stake holders.

- Availability of supporting data

The dataset is available for 24X7X365 days as per the Google Platform (GCP) provisioning in the cloud service level agreement.

- Competing interests

The data, program, artefacts are available for the all the stake holders with free of cost and there is no commercial interest.

- Funding

GCP Guild Team have contributed for the WHO dataset and creating the program for the benefit of the Covid-19 vaccination and WHO – Nature Labs has provided the GCP billing and supported for the programing, big query knowledge transfer.

- Authors' contributions

Kyndryl Solutions Private Limited, GCP Guild members have contributed the time and efforts for the WHO successful in provision the dataset and automated Python programing, Big Query Tables.

- Acknowledgements

WHO – Nature Labs research team is acknowledged the contributions of Kyndryl solutions Private Limited in creating the GCP projects and provide the data migration, Big Query and Python programing for the data analytics. Also thankful to Google for provide the Log Analytics for Nature Labs project.

- Authors' information (optional)

Kyndryl Solutions Private Limited Authors are Subject Matter Expert, GCP Cloud Platform Architect, Big Data Engineering with solution and development experts for WHO – Nature Labs Project.

About WHO the dashboard

The World Health Organization (WHO) Health Emergency Dashboard is a platform which aims to share information about public health events and emergencies. The data on the dashboard is refreshed every fifteen (15) minutes and data is accurate as at time of refreshing.

The WHO Health Emergency Dashboard is not a comprehensive representation of all the events and emergencies that WHO is aware of and responding to. The events displayed are a subset of those reported through official channels as mandated by the International Health Regulations (IHR 2005). The content of the WHO Health Emergency Dashboard is for general information only. It is subject to change without notice. While every reasonable effort has been made to make the information on the WHO Health Emergency Dashboard as timely and accurate as possible, WHO makes no claims, promises or guarantees about the effectiveness, completeness and accuracy of the contents of the WHO Health Emergency Dashboard, and expressly disclaims any liability for damages as a result of the use and/or application of the WHO Health Emergency Dashboard, errors and/or omissions in the content. The responsibility for the interpretation and use of the content lies with the reader. WHO reserves the right to make updates and changes to posted content without notice and accepts no liability for any errors or omissions in this regard. WHO assumes no responsibility or liability for any consequence resulting directly or indirectly from any action or inaction readers take based on or made in reliance on the information and material available on the WHO Health Emergency Dashboard. While every reasonable effort has been made to use appropriate language and pictures on the WHO Health Emergency Dashboard, WHO expressly disclaims any responsibility for inadvertent offensive or insensitive, perceived or actual, language or pictures. WHO will take no responsibility for or be liable for the WHO Health Emergency Dashboard being temporarily unavailable in the event of technical or other issues.

The size of the pie charts corresponds to the number of events in a given country or territory; the size does not indicate the severity or risk associated with the event(s). The designations employed and the presentation of content on the WHO Health Emergency Dashboard, including names of the events, maps and other illustrative materials, do not imply the expression of any opinion whatsoever on the part of WHO concerning the legal status of any country, territory, city or area, or of its authorities, or concerning the delineation of frontiers and borders. Grey areas on maps represent approximate border lines for which there may not yet be full agreement.

Reference :

[WHO Coronavirus (COVID-19) Dashboard | WHO Coronavirus (COVID-19) Dashboard With Vaccination Data](https://covid19.who.int/data)

The GCP drive for getting the dataset

### <https://drive.google.com/file/d/1zrm7T5abytOKQU0rLDcF7-ujcyH7CwNj/view?usp=sharing>