Project Report

## Project name: "COVID-19 Data Integration, Analysis, and Visualization Platform" Student: Sergejs Vlasovs Bootcamp: Big Data Engineering

## https://github.com/vl-sergei/accenture\_bootcamp/tree/main/Final%20Project

**Task 1. Use Snowflake Marketplace and get COVID-19 free dataset. Setup Snowflake Resource monitors.**

An account was created on Snowflake. Resource monitoring was configured using the SQL code below:

**CREATE** **OR** REPLACE RESOURCE MONITOR limiter

**WITH** CREDIT\_QUOTA = 400

TRIGGERS **ON** 40 PERCENT DO NOTIFY

**ON** 80 PERCENT DO SUSPEND

**ON** 100 PERCENT DO SUSPEND\_IMMEDIATE;

**Task 2. Data Exploration and Enhancement.**

After studying the provided COVID-19 dataset, the idea emerged to compare the ratio of deaths to cases with the number of vaccinations during the peak periods of the pandemic. The goal was to examine the impact of vaccination during peak periods on mortality rates.

A decision was made to search for additional data to complement the table COVID19\_EPIDEMIOLOGICAL\_DATA/PUBLIC/ECDC\_GLOBAL with information beyond December 14, 2020. Several datasets from various sources were reviewed. The dataset 'Coronavirus (COVID-19) In-depth Dataset' was found on the kaggle.com platform.   
The dataset was downloaded and uploaded to the storage of the AWS S3 service:  
s3://sv-web-01/dataset/owid-covid-data (1).csv

At AWS, in the IAM (Identity and Access Management) service, a role and policy were configured to enable the uploading of files to Snowflake:

* ***Policy:***
* {
* "Version": "2012-10-17",
* "Statement": [
* {
* "Effect": "Allow",
* "Action": [
* "s3:PutObject",
* "s3:GetObject",
* "s3:GetObjectVersion",
* "s3:DeleteObject",
* "s3:DeleteObjectVersion"
* ],
* "Resource": "arn:aws:s3:::sv-web-01/dataset/\*"
* },
* {
* "Effect": "Allow",
* "Action": [
* "s3:ListBucket",
* "s3:GetBucketLocation"
* ],
* "Resource": "arn:aws:s3:::sv-web-01",
* "Condition": {
* "StringLike": {
* "s3:prefix": [
* "dataset/\*"
* ]
* }
* }
* }
* ]
* }

***Role:***  
{

    "Version": "2012-10-17",

    "Statement": [

        {

            "Effect": "Allow",

            "Principal": {

                "AWS": "arn:aws:iam::926672153406:user/nqwh0000-s"

            },

            "Action": "sts:AssumeRole",

            "Condition": {

                "StringEquals": {

                    "sts:ExternalId": "XE75838\_SFCRole=2\_yLMxFkb4djczbzQXZvf9lDz8mfw="

                }

            }

        }

    ]

}

The next step involved creating integration with the storage, setting up the database, stage, and performing other necessary actions for the dataset upload:

-- *Creating integration with the S3 service*

**CREATE** STORAGE INTEGRATION S3\_int

**TYPE** = EXTERNAL\_STAGE

STORAGE\_PROVIDER = **'S3'**

ENABLED = **TRUE**

STORAGE\_AWS\_ROLE\_ARN = **'arn:aws:iam::533266996171:role/snoflake\_role'**

STORAGE\_ALLOWED\_LOCATIONS = (**'s3://sv-web-01/dataset/'**)

**DESC** INTEGRATION S3\_int;

-- Creating of DB to operate with datasets

**CREATE** DATABASE kaggle\_dataset;

*--Creating of CSV format file*

**CREATE** FILE FORMAT my\_csv\_format

**TYPE** = **'CSV'**

FIELD\_OPTIONALLY\_ENCLOSED\_BY = **'"'**

COMPRESSION = **'AUTO'**;

*-- Creating of Stage*

USE **SCHEMA** kaggle\_dataset.public;

**CREATE** STAGE S3\_stage\_01

STORAGE\_INTEGRATION = S3\_int

URL = **'s3://sv-web-01/dataset/'**

FILE\_FORMAT = my\_csv\_format;

*-- Creating the table with certain columns*

**CREATE** **TABLE** kaggle\_covid19 (

iso\_code **VARCHAR**,

continent **VARCHAR**,

location **VARCHAR**,

**date** **DATE**,

total\_cases **FLOAT**,

new\_cases **FLOAT**,

total\_deaths **FLOAT**,

new\_deaths **FLOAT**

);

*-- Copying and transforming of Kaggle's dataset*

COPY **INTO** KAGGLE\_COVID19 (iso\_code, continent, location, **date**, total\_cases, new\_cases, total\_deaths, new\_deaths)

**FROM** @S3\_stage\_02

ON\_ERROR = **'CONTINUE'**

FILE\_FORMAT = (FORMAT\_NAME = **'my\_csv\_format'**);

After the upload, both datasets were merged, resulting in a new table:

*--Changing of lengh of ISO3166\_1 from 2 to 16777216*

**ALTER** **TABLE** COVID19\_EPIDEMIOLOGICAL\_DATA.PUBLIC.ECDC\_GLOBAL

**MODIFY** **COLUMN** ISO3166\_1 **VARCHAR**(16777216);

*-- Create a new table as a result of merging*

**CREATE** **TABLE** **IF** **NOT** **EXISTS** covid19\_MERGED\_TABLE **AS**

**SELECT** \*

**FROM** KAGGLE\_DATASET.PUBLIC.ECDC\_GLOBAL;

*-- Merge operation to update and insert records into the new table*

**MERGE** **INTO** covid19\_MERGED\_TABLE **AS** target

**USING** KAGGLE\_DATASET.PUBLIC.KAGGLE\_COVID19 **AS** **source**

**ON** target.**DATE** = **source**.dae **AND** target.ISO3166\_1 = **source**.iso\_code

**WHEN** **MATCHED** **AND** **source**.**date** >= **'2020-12-15'**

**THEN** **UPDATE** **SET**

target.CASES = **source**.total\_cases,

target.CASES\_SINCE\_PREV\_DAY = **source**.new\_cases,

target.CONTINENTEXP = **source**.continent,

target.COUNTRY\_REGION = **source**.location,

target.**DATE** = **source**.**date**,

target.DEATHS = **source**.total\_deaths,

target.DEATHS\_SINCE\_PREV\_DAY = **source**.new\_deaths

**WHEN** **NOT** **MATCHED** **AND** **source**.**date** >= **'2020-12-15'**

**THEN** **INSERT** (CASES, CASES\_SINCE\_PREV\_DAY, CONTINENTEXP, COUNTRY\_REGION, **DATE**, DEATHS, DEATHS\_SINCE\_PREV\_DAY, ISO3166\_1)

**VALUES** (**source**.total\_cases, **source**.new\_cases, **source**.continent, **source**.location, **source**.**date**, **source**.total\_deaths, **source**.new\_deaths, **LEFT**(**source**.iso\_code, 3));

After obtaining the new table and conducting a more detailed examination, it became apparent that, even in its expanded form, it provides too little data for analysis, as the latest information on cases and deaths is dated May 12, 2021. Following additional scrutiny of the original dataset, the decision was made to utilize the existing table of weekly cases and deaths since its latest entries are dated the end of 2023. However, since the vaccination table contains daily statistics rather than weekly, it became necessary to convert this table from a daily format to a weekly one:

*--Creating, formating and transforming vaccination table*

**CREATE** **OR** REPLACE **TABLE** KAGGLE\_DATASET.PUBLIC.OWID\_VACCINATIONS **AS**

**SELECT**

COUNTRY\_REGION,

ISO3166\_1,

**SUM**(PEOPLE\_VACCINATED) **AS** PEOPLE\_VACCINATED\_WEEKLY,

**SUM**(PEOPLE\_FULLY\_VACCINATED) **AS** PEOPLE\_FULLY\_VACCINATED\_WEEKLY,

**SUM**(DAILY\_VACCINATIONS) **AS** WEEKLY\_VACCINATIONS,

**MIN**(DATE\_TRUNC(**'week'**, **DATE**)) **AS** **DATE**

**FROM** COVID19\_EPIDEMIOLOGICAL\_DATA.PUBLIC.OWID\_VACCINATIONS

**WHERE** **DATE** >= **'2020-12-07'**

**GROUP** **BY** COUNTRY\_REGION, ISO3166\_1, DATE\_TRUNC(**'week'**, **DATE**);

Afterward, the table of weekly cases was prepared (transformed):

*--Creating and transform of weekly cases table*

**CREATE** **OR** REPLACE **TABLE** KAGGLE\_DATASET.PUBLIC.vacc\_MERGED\_TABLE **AS**

**SELECT**

COUNTRY\_REGION,

CONTINENTEXP,

ISO3166\_1,

CASES\_WEEKLY,

DEATHS\_WEEKLY,

**DATE**

**FROM** COVID19\_EPIDEMIOLOGICAL\_DATA.PUBLIC.ECDC\_GLOBAL\_WEEKLY;

The final step involved merging both tables and making some changes and improvements as described below:

*--Merging of two previous tables*

**CREATE** **OR** REPLACE **TABLE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT **AS**

**SELECT**

VM.\*,

OV.PEOPLE\_FULLY\_VACCINATED\_WEEKLY,

OV.PEOPLE\_VACCINATED\_WEEKLY,

OV.WEEKLY\_VACCINATIONS

**FROM** KAGGLE\_DATASET.PUBLIC.VACC\_MERGED\_TABLE VM

**LEFT** **JOIN** KAGGLE\_DATASET.PUBLIC.OWID\_VACCINATIONS OV

**ON** VM.**DATE** = OV.**DATE** **AND** VM.ISO3166\_1 = OV.ISO3166\_1

**WHERE** VM.**DATE** <= **'2023-11-20'**;

*--Changing of values type*

**ALTER** **TABLE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT

**ADD** **COLUMN** DEATHS\_PER\_1000CASES\_RATIO **NUMBER**(10,1);

**ALTER** **TABLE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT

**RENAME** **COLUMN** CASES\_DEATH\_RATIO **TO** DEATHS\_PER\_1000CASES\_RATIO;

*-- Getting death/cases per 1000 ppl ratio*

**SET** DEATHS\_PER\_1000CASES\_RATIO = **CASE**

**WHEN** CASES\_WEEKLY = 0 **THEN** **NULL** -- to handle division by zero

**ELSE** (DEATHS\_WEEKLY / CASES\_WEEKLY)\*1000

**END**;

-- Add new columns

**ALTER** **TABLE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT

**ADD** **COLUMN** WEEKLY\_VAC\_per1000 **NUMBER**(20,1);

**ALTER** **TABLE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT

**ADD** **COLUMN** PEOPLE\_FULLY\_VAC\_WEEKLY\_per1000 **NUMBER**(20,1);

**ALTER** **TABLE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT

**ADD** **COLUMN** PEOPLE\_VAC\_WEEKLY\_per1000 **NUMBER**(20,1);

*-- Update values in new columns*

**UPDATE** KAGGLE\_DATASET.PUBLIC.MERGED\_RESULT

**SET**

WEEKLY\_VAC\_per1000 = WEEKLY\_VACCINATIONS / 1000,

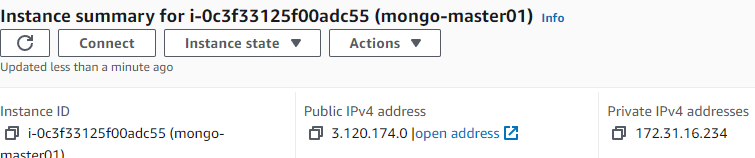
PEOPLE\_FULLY\_VAC\_WEEKLY\_per1000 = PEOPLE\_FULLY\_VACCINATED\_WEEKLY / 1000,

PEOPLE\_VAC\_WEEKLY\_per1000 = PEOPLE\_VACCINATED\_WEEKLY / 1000;

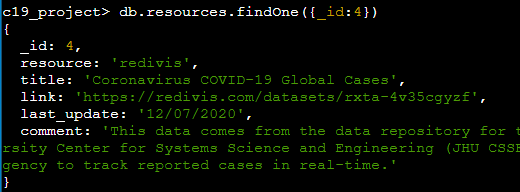
Next was proceeded with visualizing the obtained data, as described below.

**Task 3. Data Modeling in NoSQL.**

An Ubuntu server was deployed on the AWS EC2 service. MongoDB was installed on it.



The database was used to store information about all the datasets studied during the selection process. The schema is provided below:



Data insertion:

# Create a document

resource\_data = {

    "\_id": 4,

    "resource": "redivis",

    "title": "Coronavirus COVID-19 Global Cases",

    "link": "https://redivis.com/datasets/rxta-4v35cgyzf",

    "last\_update": "12/07/2020",

    "comment": "This data comes from the data repository for the 2019 Novel Coronavirus Visual Dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). This database was created in response to the Coronavirus public health emergency to track reported cases in real-time."

}

# Insert the document into the collection

result = coll.insert\_one(resource\_data)

print(f"Inserted document ID: {result.inserted\_id}")

**Task 4. API Development with Python:**

The following APIs were deployed for working with MongoDB and Snowflake:

* ***MongoDB:***

import pymongo

from pymongo import MongoClient

# Connect to MongoDB

client = pymongo.MongoClient('mongodb:// 3.120.174.0:27017)

# Create or get the database

db = client.c19\_project

coll = db.resources

client.close()

* ***Snowflake:***

# connector to Snowflake

conn = snowflake.connector.connect(

    user = 'SERGEI',

    password = changemeP1s,

    account = 'tgdbsvg-lh06896',

    warehouse = 'COMPUTE\_WH',

    database = 'KAGGLE\_DATASET',

    schema = 'PUBLIC',

    role = 'ACCOUNTADMIN'

)

cur = conn.cursor()

cur.execute('SELECT \* FROM MERGED\_RESULT')

# Storing the data into variable

data = cur.fetchall()

vac\_death = pd.DataFrame(data, columns = [x[0] for x in cur.description])

vac\_death.to\_csv('vac\_death\_data.csv', index=False)

# print(vac\_death)

cur.close()

conn.close()

**Task 5. Interactive Visualization with Python:**

For data visualization, the Dash framework is being used:

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

from dash.dependencies import Input, Output

import plotly.express as px

import pandas as pd

import snowflake.connector

vac\_death['DATE'] = pd.to\_datetime(vac\_death['DATE'])

# Create Dash app

app = dash.Dash(\_\_name\_\_)

# Define layout

app.layout = html.Div([

    dcc.Dropdown(

        id='country-dropdown',

        options=[{'label': country, 'value': country} for country in vac\_death['COUNTRY\_REGION'].unique()],

        value='Austria',

        multi=False

    ),

    dcc.Graph(id='ratio-vac-chart'),

])

# Define callback to update the line plot based on selected country

@app.callback(

    Output('ratio-vac-chart', 'figure'),

    [Input('country-dropdown', 'value')]

)

def update\_scatter\_plot(selected\_country):

    # Filter data based on user selection or other criteria

    filtered\_df = vac\_death[vac\_death['COUNTRY\_REGION'] == selected\_country]

    # Create scatter plot

    fig = px.scatter(

        filtered\_df, x='DATE', y=['DEATHS\_PER\_1000CASES\_RATIO', 'WEEKLY\_VAC\_PER1000'],

        title='Vaccination and Death cases', labels={'value': 'Cases per thousand people', 'DATE': 'Time'}

    )

    return fig

# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

    app.run\_server(debug=True)

**Post-Bootcamp actions**

The following actions were taken to ensure an opportunity for further work with the dataset.

1. Data set was saved into CSV file:

df\_pandas = pd.read\_csv('modified\_data.csv')

1. Original python code was modified to work with local CSV file
2. The Docker file was created:

FROM python:alpine

WORKDIR /project\_app

COPY . .

RUN pip install --no-cache-dir dash plotly pandas

EXPOSE 8050

CMD ["python", "project.py", "--host", "0.0.0.0", "--port", "8050"]

1. Project folder with CSV, Docker file, and python file was pushed to GitHub:

<https://github.com/vl-sergei/accenture_bootcamp/tree/main/Final%20Project/project_app>

To create the Docker image, execute the following command in the terminal from the project folder:

*docker build . -t final\_project*

To deploy the container, use the following command:

*docker run -it -p 8050:8050 final\_project*

Plots can be accessed at http://127.0.0.1:8050