EN.685.648.81.FA23 Data Engineering Project

By: Dana Andrea, Jackie Huang, Twesha Mitra, Brandon Morrow

Contents

[Project Overview 1](#_Toc152701625)

[Raw Dataset Description 1](#_Toc152701626)

[CDC Data 1](#_Toc152701627)

[S&P 500 Data 2](#_Toc152701628)

[FRED 2](#_Toc152701629)

[EPA 2](#_Toc152701630)

[ETL: Data Preprocessing and Transformation 2](#_Toc152701631)

[ERD Diagram & Data Dictionary 3](#_Toc152701632)

[Create Schema, Tables, and Load Data 5](#_Toc152701633)

[Automation and API 7](#_Toc152701634)

[Step by Step instructions to run the pipeline 8](#_Toc152701635)

# Project Overview

In this project, our team built and end-to-end data pipeline for COVID-19, economic, financial, and Carbon Dioxide emissions data. The vision for our project is a dataset that a team of data analysts and scientists can use to see how COVID-19 pandemic influenced the economy and climate. The data sources are the from Center of Diseases Control (CDC), Standard and Pools 500 (S&P 500), Federal Reserve Economic Data (FRED), and the Environmental Protection Agency (EPA).

# Raw Dataset Description

### CDC Data

<https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data/vbim-akqf/explore>

Our COVID-19 raw data is the CDC daily reporting dataset containing public use data. This data is at the national level, and contains columns such as date reported, sex, age group, ethnicity, hospitalized, icu, and death. We will join this data set with our others on the CDC report date column. This date column ranges from 2020-06-23 to 2023-10-02.

### S&P 500 Data

<https://finance.yahoo.com/quote/SPY/>

The S&P 500 data is obtained by using the Python yfinance library to access Yahoo Finance API. The data includes stock price of open, close, high, low, and adjusted close, also with the trading volume. The date column ranges from 2020-01-02 to 2023-11-26.

### FRED

<https://fred.stlouisfed.org/>

Short for Federal Reserve Economic Data, FRED is an online database consisting of hundreds of thousands of economic data time series from scores of national, international, public, and private sources. We will use this data to historic information for key economic indicators in the US including, unemployment, GDP, Real GDP, Federal Funds Rate, and interest rates. All of this data can be extracted as a .csv file. We will want to normalize all these indicators into one table or a few tables, joining them together by timeframe.

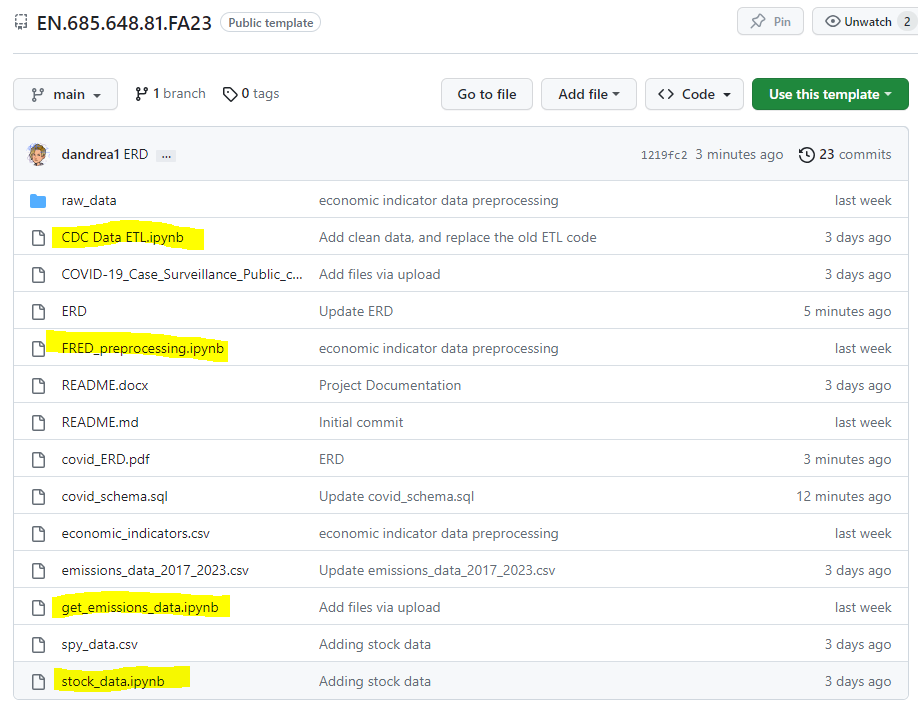
### EPA

<https://aqs.epa.gov/aqsweb/documents/data_api.html>

The EPA API provides emissions data by day for all locations in the United States. We will be combining this with COVID data on the date column to see if there was a decrease in emissions from COVID lockdowns. During our processing we take the daily national average of different air quality measures.

# ETL: Data Preprocessing and Transformation

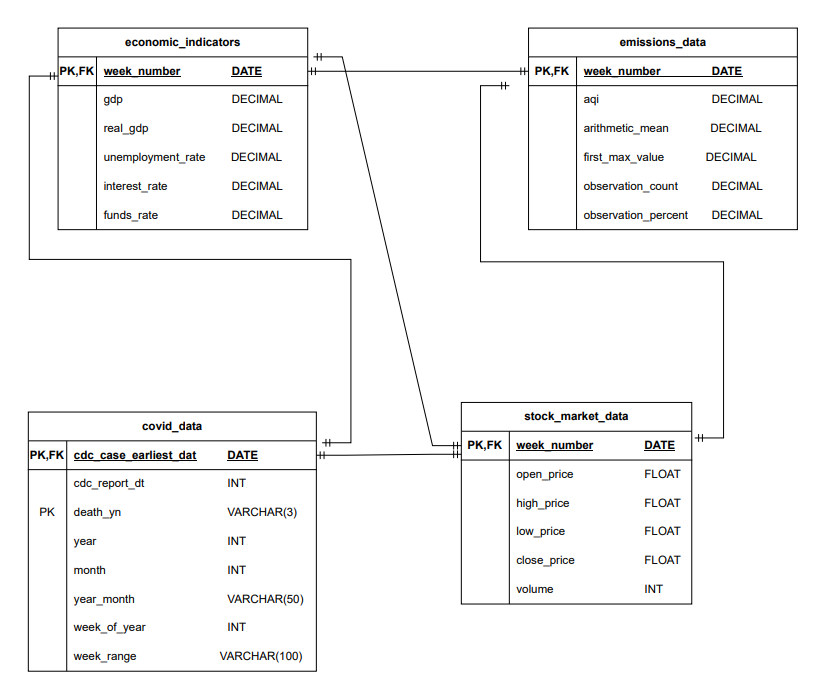
Each team member extracted a different data source and cleansed it to be ready to insert into the Postgres database. There is a total of 4 Jupyter notebooks, which are highlighted in below screenshot:



In addition to modifying the correct data types, addressing missing values, removing unwanted variables, and any other actions needed. All data was aggregated or disaggregated to the week level to be fit for our database.

# ERD Diagram & Data Dictionary

The ERD diagram is shown below to address the database design.

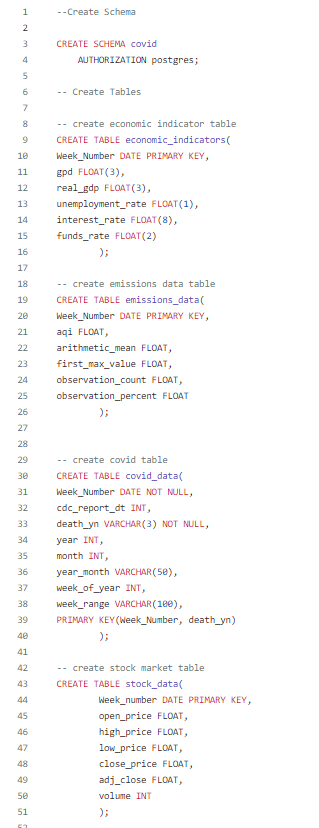


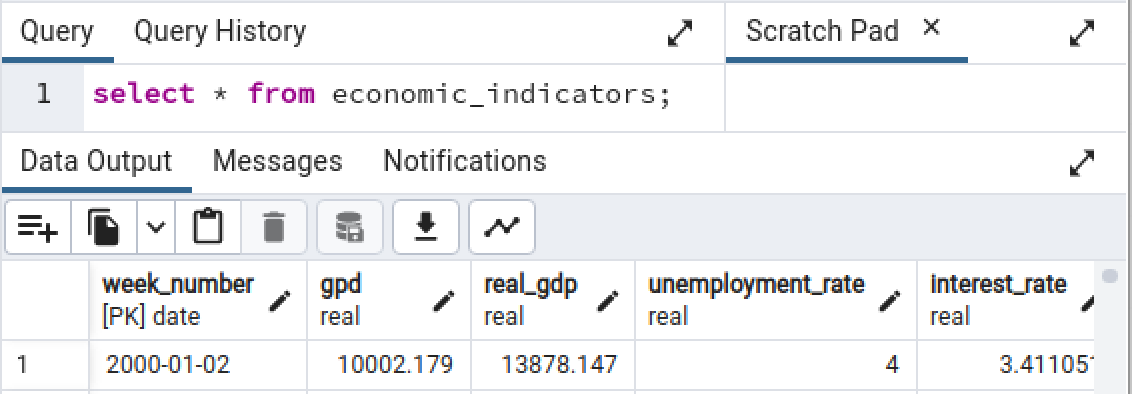
Here attached the data dictionary for each of the tables.

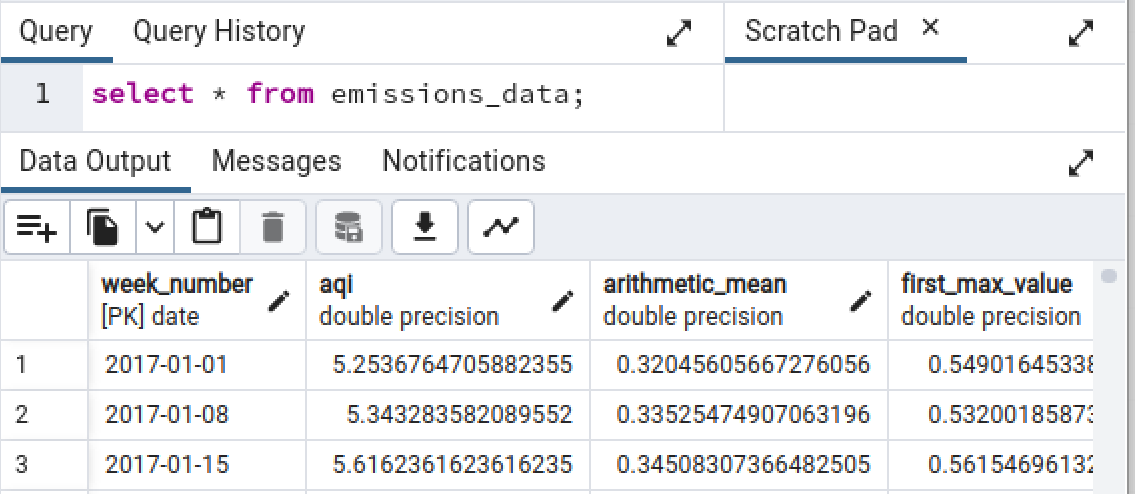
* Table: Economic Indicators
  + week\_number: represents the week start day
  + gdp: gross domestic product in billions of dollars for the USA
  + real\_gdp: real gross domestic product in billions of dollars for the USA
  + unemployment\_rate: rate of unemployment in the USA as a percent
  + interest\_rate: rate of interest (cost of borrowing money) in the USA as a percent
  + funds\_rate: federal fund rate in the USA as a percent
* Table: Emissions
  + Aqi: the overall air quality for the given day
  + arithmetic\_mean: Average pollutant concentration value for the given day
  + first\_max\_value: Average highest pollutant concentration value for the given day
  + observation\_count: total number of observations or measurements taken on the given day
  + observation\_percent: the percentage of observations that are considered valid or reliable out of the total number of observations
* Table: Covid
  + cdc\_case\_earliest\_dt: The earlier of the Clinical Date (date related to the illness or specimen collection) or the Date Received by CDC
  + cdc\_report\_dt: number of confirm cases with the specified conditions
  + death\_yn: the patient is dead or not
  + year: confirm case year
  + month: confirm case month
  + year\_month: confirm case year and month
  + week\_of\_year: the week number of the cdc\_case\_earliest\_dt
  + week\_range: the date range for the week\_of\_year
* Table: S&P 500
  + Open\_price: stock price at open time
  + high\_price: highest stock price in a day
  + low\_price: lowest stock price in a day
  + close\_price: stock price at close time
  + volume: total trading volume of the stock in a day

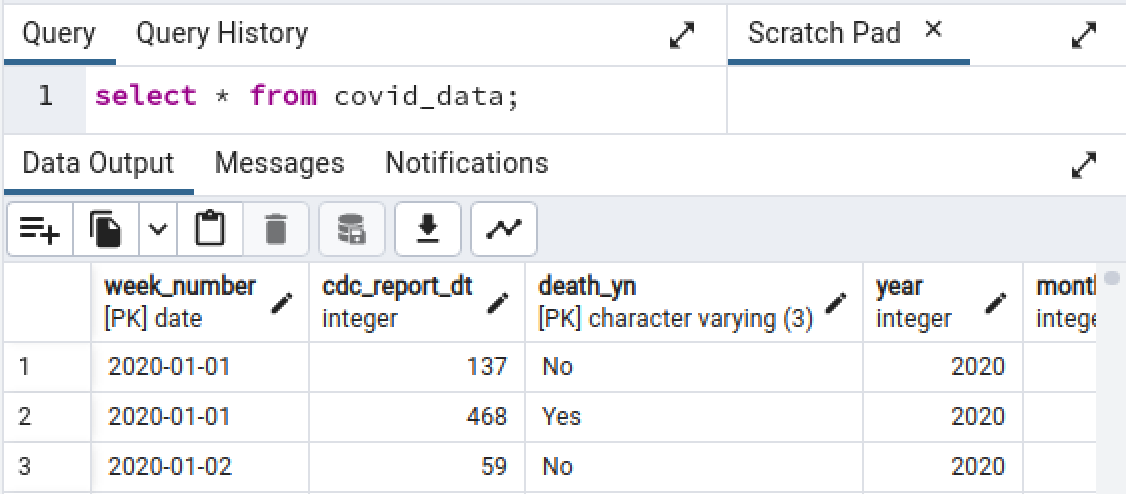
# Create Schema, Tables, and Load Data

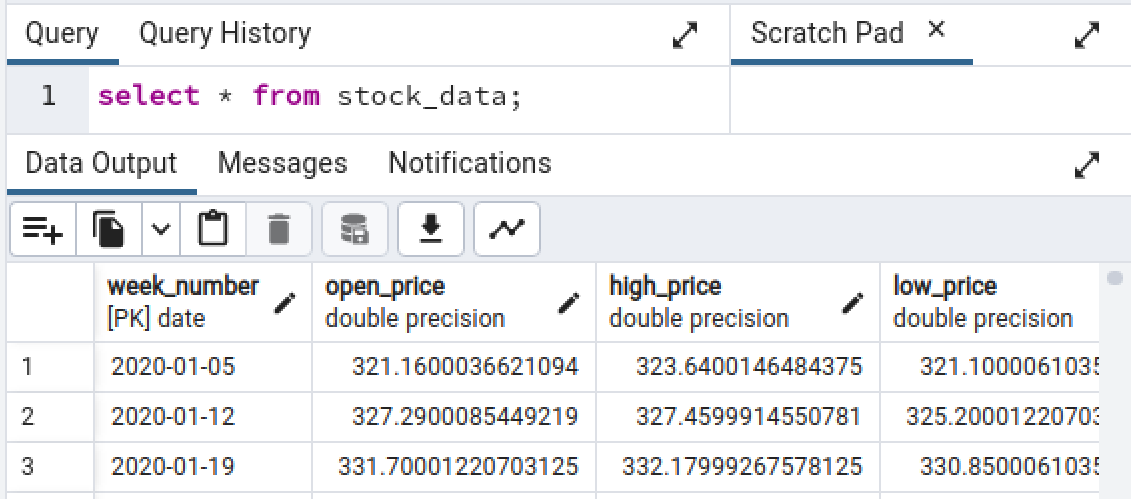
The schema and tables are created in the [covid\_schema.sql](https://github.com/dandrea1/EN.685.648.81.FA23/blob/main/covid_schema.sql) file. Also, the cleaned .csv data is loaded into the tables using the COPY method.











# Automation and API

For our automation, we built an Airflow DAG that will run Python scripts that will update our data weekly.

There are a total of 5 Python scripts. The EPA and stock data are available through API so these scripts pull the past week of data from the API and will insert it into our covid database. The CDC and Economic data are set up so that if a new csv file is dropped into the folder that include airflow scripts, then the previous weeks data will be pulled from the csv and inserted into the database. The scripts for CDC and Economic data will detect if there is an updated csv file available, and if there is no file, it simply prints “No updates for CDC data” or “No updates for Economic Data”. There is also a update\_database script to import the updated tables into the database.

In summary, the workflow for the DAG is shown below:

A diagram of a software process

Description automatically generated

This DAG executes weekly. All four data sources pull data from the previous week and then update the database. A Flask API is then started, and a report is generated in the API. After this script executes the Flask API remains open, and reports on each of the four data sources can be pulled. For example, <http://localhost:8001/api/covid> returns the COVID-19 data from the database in json format.

Our Flask API allows access to the covid schema contents, and has 4 GET functions for each of the 4 data sources. This API runs on localhost 8001, and remains running after completion of the DAG, or can be activated manually to get data from the covid schema. As shown in the instruction below, the updated data can be assessed by running API again, for example,  [<http://localhost:8001/api/covid> would return updated COVID-19](http://localhost:8001/api/covid%20would%20return%20updated%20COVID-19) data, if there is any update. Otherwise, API will output “No updates for CDC data”. In addition, we create 3 charts to visualize some of the key variables relationships, such as week vs number of confirmed COVID-19 cases.

# Step by Step instructions to run the pipeline

1. Open and login to the class Virtual Machine.
2. Download the project zip\_file to your Downloads: **<name of file>**
3. Extract the files.
4. Open Postgres and copy paste the [covid\_schema.sql](https://github.com/dandrea1/EN.685.648.81.FA23/blob/main/covid_schema.sql) file into the query box.
5. Run the query. The schema and database should be created and the data loaded (see above screenshots).
   1. *Note to the reader: If the .sql file does not run due to permissions error. Follow the solutions in this stack overflow:* [*https://stackoverflow.com/questions/50273803/postgresql-could-not-open-file-for-reading-permission-denied-sql-state-42501*](https://stackoverflow.com/questions/50273803/postgresql-could-not-open-file-for-reading-permission-denied-sql-state-42501)
6. tbd

# 