Extract Transform Load Project

Team: John Chan, Emory Mansuetti, Khank-Linda Stark,

Jackson Freese, DeJuan Hall, Siddharth Das

UC Davis, Data Analytics Bootcamp

June 10, 2021

Abstract

Data literacy has become an essential skill for business managers and professionals in all fields. The ETL project is a case study that employs technical capabilities that we have learned in class. We wanted to compare how COVID-19 impacted to two coastal states – New York and California. In order to do this, we had to extract COVID-19 data from three disparate sources, transforming the data, e.g., performing data cleaning, concatenation, and aggregation, to suit business requirements, and finally load the data into a database that can be used for future analysis or business use.

Keywords: ETL, Extract, Transform, Load, Python, SQL, APIs, pgAdmin 4, iFrame

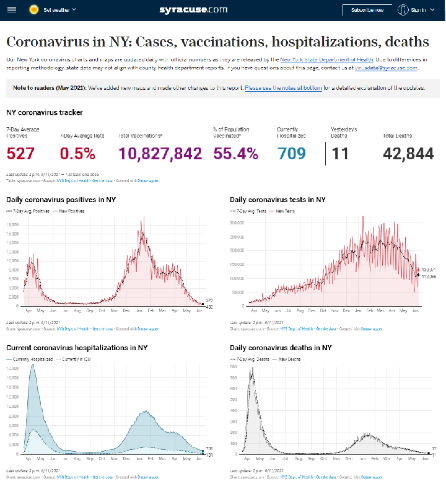
Extract Transform Load Project

# Proposal and Preparation

The team discussed the fact that we had a little over a week to accomplish our work. We agreed to start with modest goals that would allow us to complete the project earlier than the allotted time, which would allow us to work on more complex tasks if we had time.

## Extraction1

We initially agreed to use two different datasets from, “The COVID Tracking Project”: a CSV file and data extracted from an API. However, we found that we had extra time, so we took on an additional challenge of scraping dynamic chart URLs, embedded in iFrames, on the Syracuse website.



We selected one of the URLs and loaded it into a browser. We then inspected the page, looking into the Network, then examined if there were XMLhttpRequests (XHR). We found that there was another URL, which led us to datawrapper.de, a provider of dynamic visual wrappers.

Graphical user interface, chart

Description automatically generated

We then downloaded the chart’s CSV data. – our third dataset.

* The CA CSV data was directly downloaded from the website
* JSON API was used to request services and retrieve data for the NY dataset
* For the Syracuse URL, urllib.request was used to fetch and read the website
* Beautifulsoup parsed html page, found src URLs embedded in iFrames
* We selected one URL, “static.dwcdn.net/data/ijEiy.csv?v=1623222240000”, a dynamic chart, loaded it into a browser, used the inspect tool to peer into the Network activity. We then filtered by XHR to find URL redirect.
* Python and Response were used to extract, then export the chart data to CSV.

## Selected Data Fields:

|  |
| --- |
| Data Field Name |
| Date (Primary Key) |
| state |
| deathIncrease |
| hospitalizedIncrease |
| inIcuCurrently |
| PositiveCasesViral |
| positiveIncrease |
| totalTestResults |
| totalTestResultsIncreas**e** |

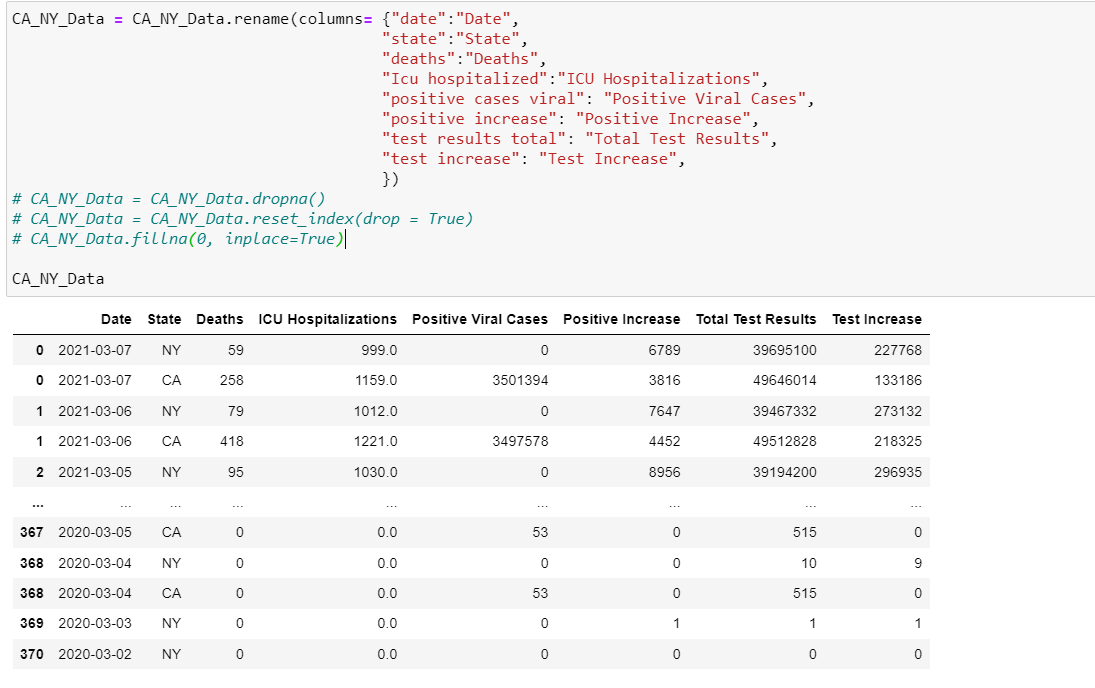
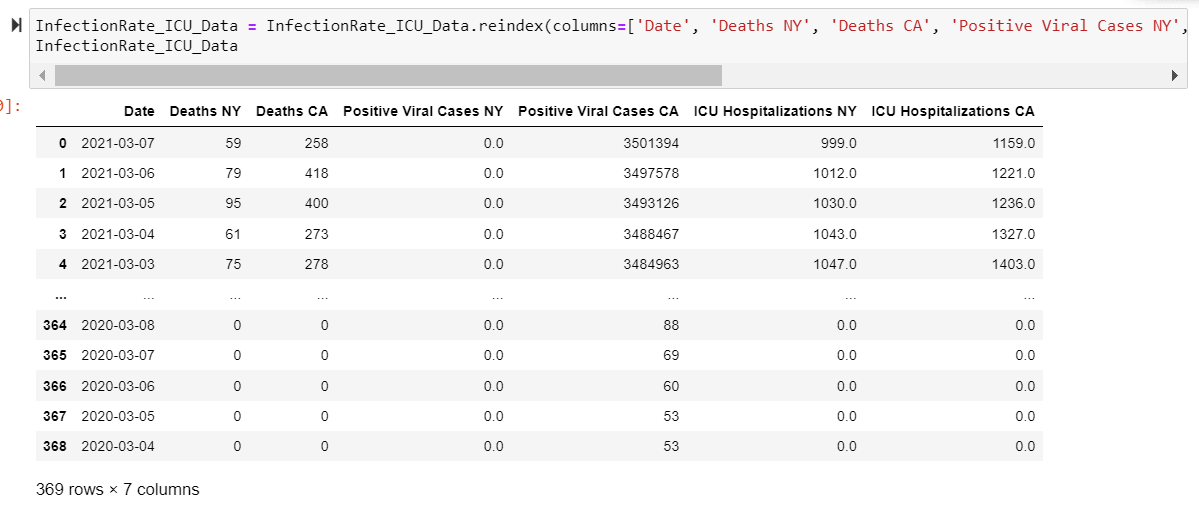
## Data Sources:

* <https://api.covidtracking.com/v1/states/ca/current.csv>
* <https://api.covidtracking.com/v1/states/ny/daily.json>
* <https://www.syracuse.com/coronavirus-ny/>
* <https://datawrapper.dwcdn.net/ijEiy/2/>
* <https://static.dwcdn.net/data/ijEiy.csv?v=1623222240000>

## Transform

We performed transformation of the data to suit our needs, including:

* Pandas functions used:
* CA: Load CSV file, drop unnecessary columns, renamed NaN/missing values to zero, and exported to back to CSV
* NY: Dictionary of variables created to store data from the requested API calls
* Converted dictionary into DF, changed date format, exported to a CSV file
* Performed concatenation merge, inner of NY and CA datasets
* Removed “totalTestResultsIncrease” column, as it contained zeros
* Renamed columns to make it more readable
* Sort the merged DataFrame for each item listed below, and Paste image of dataframe.head().
* Deaths
* Daily Hospitalizations
* ICU Hospitalized



# Load

Both the CA and NY datasets were transformed into DataFrames that were subsequently loaded into to the SQL database on Postgres, using an ipynb file that was created specifically to load the datasets. We then used the Quick Database Diagrams website to create the database table schemes that were then loaded into the Postgres database.

Graphical user interface, application

Description automatically generated

# Summary

The project went well. We ran out of time to compare how COVID impacted two coastal cities for our presentation. However, we learned a few new things and encountered some challenges:

1. There was a learning curve on how to do a proper fork of the central repository
2. Some of the data had floating point decimals, so the database scheme had to be changed to a float type instead of int type.
3. The was a GitHub merge conflict that was resolved by examining the code, using git status to see where the conflict was, then using the git mergetool to remove the offending line. The file was then saved. Next a git rebase –continue to apply the commit that was being pushed to the central repository.
4. Encountering iFrames was something new that was not covered in class, so it took a little bit of Google-Fu to figure out how to use the web page inspect tool and filtering the Network XHR to find where to extract the data.