ETL Project Report

Objective:

To track the coronavirus cases in Italy, China, South Korea, and the US in the last 3 months, as well as the main stock market index for each nation on a day-by-day basis. With this data we will construct a sql relational database. With a cleaned and structured database, we can observe trends and predict future market behavior. We will also add in an additional database which tracks key events during this pandemic and see their impact on both the market and case counts.

Extract

Data sources

1. Infection data from a github repository managed by Johns Hopkins CSSE, which has daily tracking of Covid 19 cases around the world. They have a variety of sources such as WHO, USA CDC. Data comes in the form of csv files of each day's infection data. https://github.com/CSSEGISandData/COVID-19
2. Market Index Data from https://www.investing.com/ where we grab the primary market index for each respective nation. DJI for US, SSE composite for China, FTSE MIB for Italy and KOSPI for South Korea. Data come in CSV files for each index in the last 3 months
3. Grab major event data by web scraping this CNN article https://www.cnn.com/2020/02/06/health/wuhan-coronavirus-timeline-fast-facts/index.html And stored as a csv file.

Raw data are in the raw_data folder of the repository

Transform

Clean data by

- Removing unwanted data, such as market open price, recovery patient numbers
- Modifying data type to sql and manipulation friendly format such as Date and floats instead of strings
- Unify data format, such as making all dates into ISO format, condense country names e.g. [South Korea, (Korea, South), Republic of Korea] into just South Korea
- Rearrange and combine dozens of raw data csv files into 6 csv files, one for each table in our end sql database. One for each of the four country, one for a total table and one for a major event table

Cleaned data stored in the clean_data folder

Cleaning scripts in the data_cleaning script

Load

Since most of our raw data were in tabular formats already, a relational database made the most sense.

Using panda dataframes, sqlite, sqlalchemy, we loaded the csv data files into a relational database where the date is the key that connects each table. Each table's date column is also their primary key

We have 6 tables total

1. US_data
2. China_data
3. Italy_data
4. Korea_data
5. Total_table
6. Event_table

Database is stored in a sqlite file
Also loaded the files into pgadmin as an alternative method
Table schemas and screenshots are in the pgadmin_files folder

Extras

Visualization

The results were visualized on the dashboards by each country. The dashboards are based on Chart.js, an open source HTML5 based JavaScript charts library. Data on the dashboards are sourced from auto-updated csv files, the process of which is described below.
Set up as a github page: https://ccchiang92.github.io/ETL-project/

Auto-Update

A extra feature where we can update our whole database and tables with the newest data just by running a couple scripts in command line

- How to use
    1. do a fresh git pull, it will not only pull this repo but also the Johns Hopkins CSSE repo for new raw csv data
       *If cloning use recursive clone
    2. Run python script auto_update.py

  And now all the data is cleaned and updated to the most recent day from our sources
  Will not work if data format/structure of sources change or if data is more than a month old. Can only update a maximum of one month's worth of data.
  Auto updated data are stored in cleaned_data\Auto_Updated for csv files
  And sqlite_database\Auto_Updated for the sqlite file
- How it works
    1. Johns Hopkins CSSE data is set up as a git submodule in raw_data so every new pull of the repo will also pull updates from them
    2. auto_update.py combines all of our ETL scripts, with some modification, from data cleaning to web scraping and sqlite construction into one file. Different steps are split into different functions.
       Investeting.com's market index data is treated differently. Instead of manually downloading a csv file for each index. The website is scraped for the latest tables. Limitation is that the scraping script is simple and can only grab upto a month's worth of data.

Auto-updated data are kept different from our manual data to prevent bugs from corrupting the whole data set

Potential Improvements

For sqlite, it rebuilds the whole database each time, Ideally it would only need to update new entries
Set up a github bot to auto run the scripts every night

Group Members