ETL Project Reflection
By: Cooper Bebeau (xqt5sy)

For this ETL project, I built a pipeline that integrated U.S. Census population estimates from a local CSV file with COVID-19 vaccination data pulled from a CDC public API. The goal was to analyze how vaccination counts compared with total population across U.S. states from 2020 to 2022. Overall, the project helped me practice real-world data handling, but there were several technical challenges I had to work through.

The first major challenge I faced involved aligning the two datasets, which used different naming conventions, identifiers, and formats. The local CSV contained population estimates with full state names, while the API returned vaccination counts using state abbreviations. To merge the two sources accurately, I created mapping dictionaries to translate between the two formats. I also had to clean and filter the data by removing entries like national totals or regional groupings that did not represent individual states, since keeping them would have distorted the analysis.

Handling the cumulative nature of the vaccination data also required extra logic. The API provided total completed vaccinations per year. To determine how many people got vaccinated in a given year, such as 2022, I needed to subtract the total from the previous year. I wrote code to compute these year-over-year values and validated the results to ensure that they did not contain negative numbers or missing data. Structuring the final output in a way that presented meaningful insights required multiple revisions. For example, summarizing the top states by vaccination count and calculating correlations between total population and vaccination levels both needed careful formatting to be readable and informative.

There was also a technical issue when I tried using MySQL to store the final dataset. I ran into several problems during the local setup, including permission errors and instability in connecting to the database. Given the time constraints and the limited scope of this project, I decided to switch to SQLite. This choice worked much better in practice, as it required no additional setup and integrated easily with pandas. It also allowed me to store a complete, queryable version of the cleaned dataset without the need for an external server.

Some parts of the project went more smoothly than expected. Once I separated each part of the pipeline into its own function, the process became much easier to manage and test. Fetching and cleaning the CDC data using chunked API calls worked well with a loop-based pagination strategy. Converting between JSON and DataFrame formats using json_normalize also worked reliably. I found it useful to be able to make quick, flexible transformations such as dropping or renaming columns and reshaping the data into a pivoted form for analysis.

By the end of the project, I had gained a deeper appreciation for the complexity involved in working with public datasets. Merging data from different sources often requires a lot of cleanup and transformation, especially when formats and meanings do not line up perfectly. This project also reminded me how important it is to think ahead about how data will be used or interpreted once it is processed.