Some of the primary challenges we faced while working on this data project dealt with reading and converting the file to a different format based on user input. To ensure that the data processor could handle multiple file types seamlessly, we experimented with various solutions and went through a process of trial and error. Specifically, we tested out various methods for distinguishing whether the input is a URL or local file and adjusting the code to detect the file format correctly. Furthermore, we faced a few challenges with converting the file to another format. Since we converted the input file to a dataframe in our initial code block, we were unsure how to properly convert and save it as a different file format. When we converted the file to a different format, it was still showing as a DataFrame type instead of being stored as a new file. As a result, we had to experiment with different ways of writing the code to ensure the data was converted and stored properly. With SQL, we had to figure out how to implement a few more steps, including establishing a connection, committing changes, and closing the connection.

Summarizing information from the DataFrame, including data file ingestion and post processing data, proved to be easier than we expected. Additionally, modifying the number of columns was not as difficult as we had anticipated. We chose to drop the last column of the DataFrame instead of adding one or dropping a specific column, making the code more adaptable to different file types. Lastly, converting the files to DataFrames helped us a lot throughout the process and enabled us to transform data more efficiently.

A utility like this is extremely valuable for future data projects. The ability to quickly convert between file types allows for flexibility in how data is processed and stored. For example, when working with APIs or URLs, utilizing a JSON format may be more useful. In other cases, such as handling large datasets with complex queries, SQL may be a more efficient data format. Storing data in SQLite is especially helpful for large datasets. The data information summaries also allow for a more user-friendly experience and simplified presentation of key aspects of the datasets. Overall, having a general-purpose data processor to convert files will be applicable and convenient for future data needs.