1. **How would you partition the data?**

The following assumes we are storing this data in file format, and not in a database, to maintain the raw data as close to the original as possible.

Would use a horizontal partitioning strategy, where each partition holds a specific data subset, each with the same schema definition – in this case data related to a particular ticker symbol.

Federated S&P 500 Data

ABT

ACN

ADBE

…

…

This is largely due to this strategy readily facilitating indexing and lookup of specific ticker-related data. The fact that this is essentially time-series data could lend itself to partitioning by date instead. However, there is not a lot of data per ticker, so by-date lookups would likely not be impacted performance-wise.

1. **Which file format would you output the data in?**

Could store the data as parquet files on Databricks (for example) for optimised storage and retrieval, but more likely to store them as csv files, as the files are individually small, and are unlikely to grow significantly over the long term. Whilst it is possible that there may be changes to the file schema in the future, this is of low probability as this type of data has been standardized for many years now.

Additionally, the file names would be standardized to include the relevant start and end dates, as well as the ticker symbol for which the data relates.

1. **What were your considerations while deciding on the partition strategy?**

Performance – can access multiple data subsets (partitions) in parallel.

Portability – able to load this raw data easily into a database or Python script.

Changes to S&P 500 component companies – If there are changes to the constituent companies that comprise the S&P 500, these can be easily made by ignoring existing ticker data and adding new ticker data simply by ignoring or creating partitions.

Individual monitoring – Can also load individual ticker data much more easily as they are separate partitions.

Manual checking – easily able to find and check out specific data by ticker by searching alphabetically through the folders, and then by date within the files.

1. **What would be your error handling strategy?**

All failed data downloads would be skipped, and an indicator that this occurred to be logged to a log file. An email could also be sent to the relevant employee to alert them that some data was not updated.

Note that the existing data for the specific affected ticker would not be over-written.

1. **What would be the benefits of using multithreaded / multiprocessors techniques and how would you implement them?**

Would allow parallel processing of files across multiple nodes. Could be implemented by either running separate jobs that run downloads according to pre-specified alphabetic ranges eg: A-E, F-J etc, or by leveraging a platform running a Spark engine, to automate the allocation of jobs among the available nodes.