SCHOOL OF DATA
by Springboard

Data Engineering Career Track
Guided Capstone Step Three

Step Three: End-of-Day (EOD) Data Load

Now that you've preprocessed the incoming data from the exchange, you need to create the final data format to store on the cloud. The cloud will also store historic exchange data, so Spring Capital can look up any trading day and easily find historic data.

This preprocessed data will be used in the following ETL process, as well as for adhoc user queries.

At the end of the last step, you have created three partitions under output_dir. It's easy to go through them one by one and create corresponding datasets. Note that the target dataset should have the specific schema required by the partition.

Learning Objectives:
By the end of this step, you will be able to…

- Create Spark DataFrames using Parquet files
- Perform data cleaning using Spark aggregation methods.
- Use cloud storage as output of Spark jobs.

Prerequisites:

- PySpark: read multiple Parquet files into a single DataFrame, transformations, write DataFrame.

3.1 Populate trade dataset

3.1.1 Read Trade Partition Dataset From It's Temporary Location

```
trade_common = spark.read.parquet("output_dir/partition=T")
```

3.1.2 Select The Necessary Columns For Trade Records

The temporary data that you get is associated with a common schema, fitting both trade and quote events. Since you're going to produce trade and quote data separately, you need to remove unnecessary columns to save space.

```
trade = trade_common.select("trade_dt", "symbol", "exchange", "event_tm", "event_seq_nb", "file_tm", "trade_pr")
```

3.1.3 Apply Data Correction

In the exchange dataset, you can uniquely identify a record by the combination of trade_dt, symbol, exchange, event_tm, event_seq_nb. However, the exchange may correct an error in any submitted record by sending a new record with the same uniqueID. Such records will come with later arrival_tm. You must ensure you only accept the one with the most recent arrival_tm.

This operation requires aggregation in order to group the dataset by their unique ID, so that a single function can be applied to a certain group entirely. In Spark, we use groupBy operation to achieve this.

```
trade_corrected = applyLatest(trade)
# [implement "applyLatest" method  to dataframe trade]
```

3.1.4 Write The Trade Dataset Back To Parquet On Azure Blob Storage

```
trade_corrected = "2020-07-29"
trade.write.parquet("cloud-storage-path/trade/trade_dt={}".format(trade_date))
```

3.2 Populate Quote dataset using the same method

3.3 Summary

In this step, you practiced the data normalization process and using cloud storage with the Spark output.

Open questions:

- If you want to run SQL queries against trade and quote data on Azure, how would you do that?

Submit this assignment:

- Commit and push the updated code to Github and submit to your mentor for review.