Utilization of GCP to Manipulate, Visualize and Analyze Data: A High Level View of S&P500 Securities

June 2020

1 Introduction

Google Cloud Platform (GCP) offers unparalleled data manipulation in comparison to other cloud vendors. In this demonstration, we take on the role of an amateur investor and operate on a small snapshot of the S&P500 Index to demonstrate GCP's ability to load, transform, and visualize data. In addition, we utilize a basic machine learning model to extrapolate on some hypotheses made to guess whether certain industries may be growing based on how other industries within its sector are performing, utilizing popular industry ETFs as our inputs.

2 Background

2.1 Motivation

Financial markets are a cornerstone of the global economy in our modern world. Vast amounts of capital and equity flow through these markets on a daily basis, and it's no exaggeration that trading on the market has billions, if not trillions, of dollars at stake. With stakes like these, it's no wonder that market investors place a lot of emphasis upon gathering information from market pricing and taking actions based on that information, oftentimes under time pressure.

GCP allows investors to easily intake information, process it, and then visualize it in an efficient manner. In addition, the data can then be used to inform investment strategies.

2.2 Basic Trend Analysis

One relatively common investment strategy is that of using *trend analysis* to decide where to invest. Investors invest based upon evaluations of the movement of the markets.

Most trend analyses are done using the price points for a security's opening and closing price, as well as its high and low that day. In addition to these basic metrics, we utilize the following additional metrics in this analysis:

- The **moving averages** over a short and long term, which are averages of a security's closing price over a period of time. These metrics smooth the unpredictable short term fluctuations of the market. We'll use this to visualize **golden crosses** and **death crosses**, which can be indicators of a reversal.
- **Supports** and **resistances**, which equate to a "floor" and "ceiling" price for a given security. Continued trends in these metrics can act as signals for where a security may be headed.

2.3 Datasets Used

For this demonstration, we utilize publicly available historical data from Yahoo! Finance. For the first portion, we utilize a file with the S&P500 Index's (^GSPC) historical data from the past 5 years. In addition (and for the second portion), we've chosen some Fidelity sector ETFs to use; the energy sector (FENY), the materials sector (FMAT), and the industry sector (FIDU). These ETFs will also be taken over 5 years.

The data is in CSV (comma-separated-values) format, and takes the following format (split into two rows here to ease reading):

Ticker	**Date**	**Open**	**High**	**Low**
^GSPC	2015-06-17	2097.399902	2106.790039	2088.860107
Close	**Adj Close**	**Volume**		
2100.439941	2100.439941	3222240000		

2.4 Notes

This demonstration is focused on GCP's capabilities; as such, the datasets we've utilized are rather small and simple. We note that GCP's limits are far beyond the $\sim$ 100 KB files we're utilizing here, and could feasibly also handle much larger datasets. Your own datasets may not match completely; however, the power of GCP allows us to be flexible with the data format we input.

Our datasets have "adjusted closing price" and "volume traded". While these are certainly useful in actual financial situations (especially with "volume traded" as an indicator of momentum for a stock), we will forgo their use in this particular demonstration.

Google Cloud Storage acts as our source of data in this demonstration, but one could feasibly connect to other services in the data lake, such as Cloud Pub/Sub. In addition, multiple alternative methods exist to handle datasets, such as uploading directly to BigQuery, or utilizing Dataflow to handle streaming data. Choose which strategy most makes sense for your organization.

3 Demo

3.1 Setting Up

Before the presentation of this demonstration, we've laid some preliminary groundwork. We've enabled all the necessary APIs for clean parsing and extraction of data, and uploaded our datasets to Cloud Storage. We've also gone into IAM and given our Compute and Dataflow service accounts BigQuery admin roles, to allow them to create and modify new tables from the ingestion process.

In addition, we've initialized an empty dataset as a location for the Dataflow pipeline. Note that subsequent uploads will overwrite this table if the same name is used, but for our purposes this will prove adequate.

3.2 ETL using Dataflow

Google Dataflow is a service that's dedicated to processing data in a pipeline for ETL operations. While its most salient use is in real-time streaming analytics, Dataflow can also perform batch processing. To aid with the pipeline's setup, Google provides many ready-made templates. In this demo, we will be using a ready-made template to execute this Dataflow pipeline (see Fig. 1).

The template we will be using is a simple Cloud Storage Text to BigQuery pipeline, which requires a couple of different parameters outside of the ones that we've given (see Fig. 2).

Firstly, it requires a BigQuery schema in JSON format to map this data on to. This schema will be fairly simple; we'll use most of our previous headers, add a couple of new fields, and create the rest of the schema through using BigQuery's powerful data exploration.

```
{
  "BigQuery Schema": [
    {
      "name": "ticker",
      "type": "STRING",
      "mode": REQUIRED
    },
    {
      "name": "date",
      "type": "DATE",
      "mode": REQUIRED
    },
    // Similar stuff above, omitting for length
    ]
}
```

Listing 1: BQSchema.json

Figure 1: Under the Dataflow service, select "Create Job from Template".

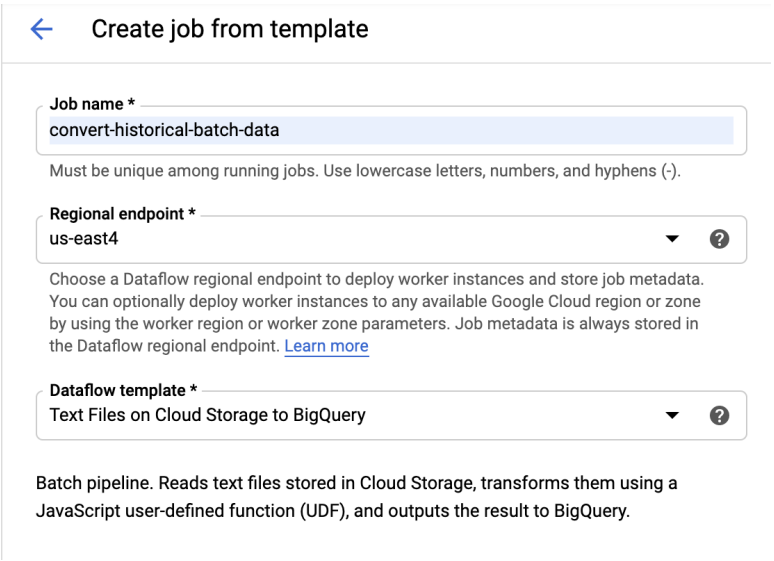

Figure 2: We choose the bulk version, since we're dealing with a static dataset.

The other main asset we need is a function that our pipeline will use to parse each given line. This function will be written in JavaScript, and implementation is also fairly simple. We can correct the data types either here or in BigQuery; here, we choose to correct them within BigQuery.

```
function transform(line) {
    var values = line.split(',');

    var obj = new Object();
    obj.ticker = values[0];
    obj.date = values[1];
    obj.open = parseFloat(values[2]);
    // similar stuff...

    return jsonString;
}
```

Listing 2: translation.js

We add these two assets to the root directory of our bucket (Fig. 3).

We're now all set to send our data through the Dataflow pipeline. We fill in all parameters as in Fig. 4, and allow the job to run for a bit. Once it finishes, our table should be populated with the requisite information.

3.3 Using BigQuery DML and Google Data Studio to Transform and Visualize Data

Now, our data is within the BigQuery table. How do we play with it?

The first thing we should do is to take a look at our data. In our initial definition of the schema and the Dataflow piping, it appears that much of our data is somewhat dirty in that they are not cleanly rounded decimals; we'd like to clean up those trailing portions. In addition, we do not define many metrics that are useful when performing trend analysis, so we should define them here.

BigQuery offers powerful tools to manipulate and wrangle data, all built within an accessible console using SQL syntax. For example, the following query does simple data cleaning by dealing with the pesky trailing digits in our original dataset:

Listing 3: "Basic Rounding Query"

```
UPDATE `booming-rush-280614.stock_data.tickers`
SET
    open = ROUND(open, 2),
    close = ROUND(close, 2),
    high = ROUND(high, 2),
    low = ROUND(low, 2)
WHERE TRUE
```

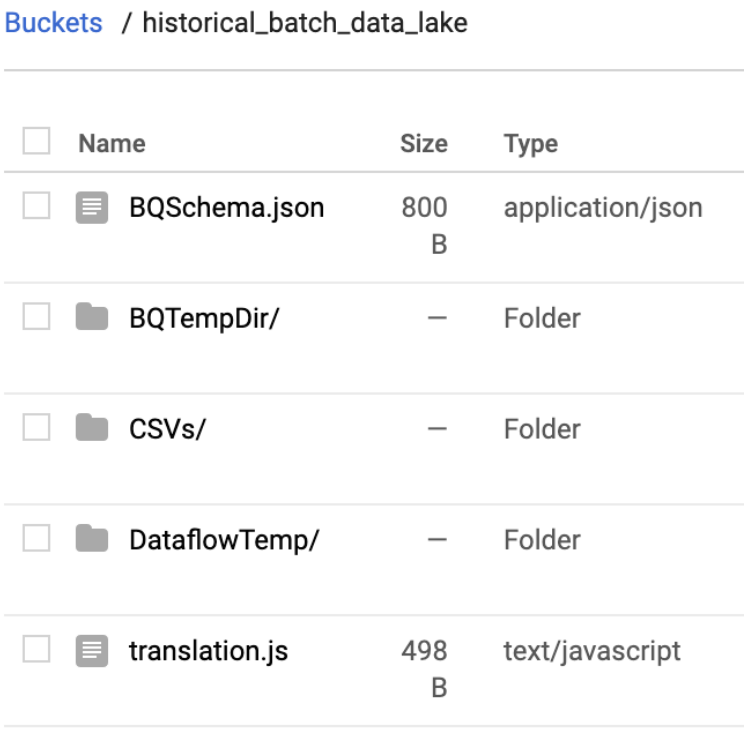

Figure 3: Included assets are in the root directory; note the addition of a temp directory.

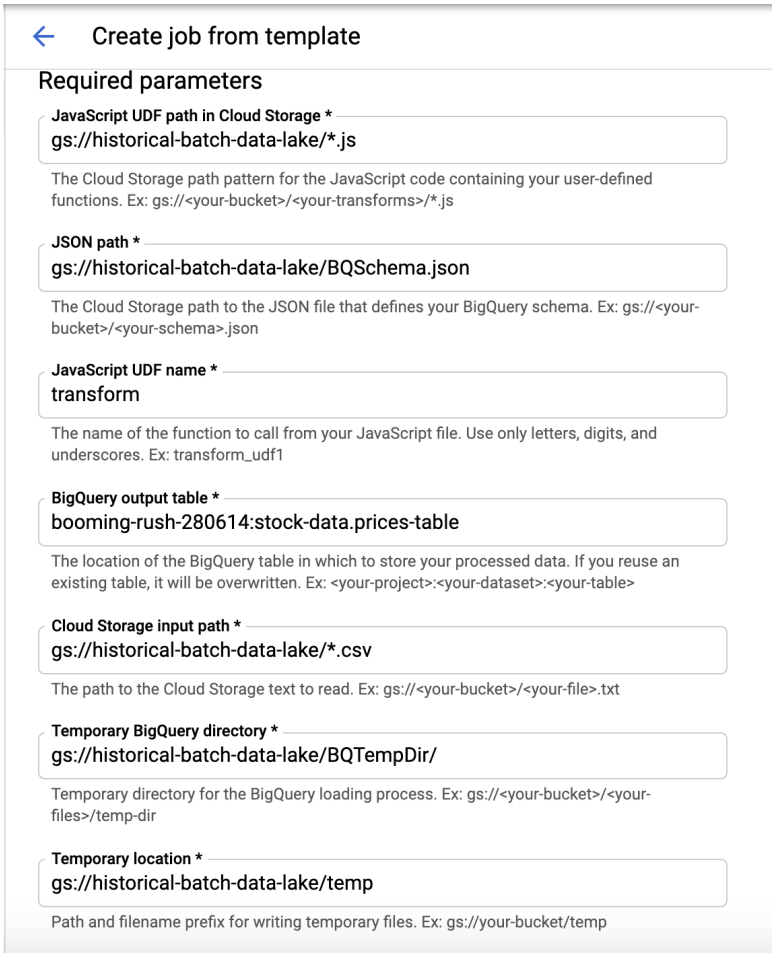

Figure 4: Your parameters may be different.

As mentioned above, many metrics such as "change", "variance", "moving average", etc. are not in our table yet. One way to add the new metrics we'd like to define would be to manually define new columns in our schema and then use BigQuery DML to fill those columns. BigQuery's data manipulation supports most standard SQL functions, and we can use those to update our table:

Listing 4: "Extracting Change/Variance"

```
UPDATE `booming-rush-280614.stock_data.tickers`
SET change = (SELECT close-open),
variance = (SELECT high-low)
WHERE TRUE
```

Another method is to run a singular request and then save the result of that request to a new table that we'll continue to build off of. This is particularly effective when running analytic functions that consolidate numerous data points into a single metric. Getting moving averages is a prime example of such a use:

Listing 5: "Extracting a 5 Day Moving Average"

```
SELECT ROUND(
  AVG(close)
    OVER (
      PARTITION BY ticker
      ORDER BY date
      ROWS BETWEEN 5 PRECEDING AND CURRENT ROW
    ), 2)
  AS five_day_average
FROM `booming-rush-280614.stock_data.tickers`
```

SQL statements (and BigQuery queries by extension) can be combined to do many operations at once as well; this works extremely effectively in tandem with the second method above to succinctly and easily wrangle data for further visualization.

Listing 6: "Mass Wrangling into New Table"

```
SELECT ticker, date, open, close,
low, high, change, variance,
  FIRST_VALUE(low)
  OVER (
      PARTITION BY ticker
      ORDER BY date
      ROWS BETWEEN 1 PRECEDING AND CURRENT ROW
    ) as previous_low,
    FIRST_VALUE(high)
  OVER (
      PARTITION BY ticker
      ORDER BY date
      ROWS BETWEEN 1 PRECEDING AND CURRENT ROW
```

```
    ) as previous_high,
-- continued...
FROM `booming-rush-280614.stock_data.tickers`
```

All of these queries can easily be run in the console's inbuilt SQL editor. In this particular demo, after running the queries necessary (in the text file attached), we end up with a table with the following schema:

- tickers and dates, which we'll use to filter the given data
- Trend magnitude, which subtracts the five-day MA from the fifteen day moving average to gauge how strong a trend is
- Booleans lower_support and higher_resistance, which show when a stock breaks a previous high or low.

We can use Google Data Studio to bring more granularity to our exploration. For example, let us take the S&P500 Index's data from June 2015 to November 2015. This coincides with the 2015-2016 stock market selloff, with major losses in worldwide markets over the period of August 21-27.

We can analyze this by selecting a chart with a time axis, and then graphing the trend direction by selecting it.

We can see by graphing the trend magnitudes that the stock market had actually been in the midst of a slight downward trend weeks beforehand, and that the selloff on Friday that continued on Monday could be noticed a couple of days beforehand, allowing investors to cut their losses if they were savvy.

3.4 Training Models using AutoML/Google AI Platform

We'll utilize AutoML tables for the next portion. GCP makes this really easy: we simply import a dataset, then set the target column to predict from the features. Here, we import the trending directions for the material and energy sectors in a new table, and try to use those to predict how the industry sector will move. Creating and deploying the model is simple; simply follow the console commands as needed.

4 Troubleshooting

As with all troubleshooting, reading the logs is immensely helpful when dealing with problems.

4.1 Dataflow

4.1.1 My Dataflow job instantly fails!

Check the parameters of the job. It's possible that the job is failing because it doesn't know where to find certain resources.

4.1.2 My Dataflow job hangs when trying to do thing with BigQuery!

This is likely a problem with user permissions. The default Dataflow service account does not inherently have the permissions to change BigQuery tables. Remember to set BigQuery roles on the service account to circumvent this.

Alternatively, check the formatting on the schema and translation functions. Header rows should not be within the function or CSV being passed, and schemas must have matching data types with the parsing function. For example, do not try to pass a string to a numeric function.

4.2 BigQuery

4.2.1 My queries don't seem to be working!

Check the parameters of your query. Most query bugs arise from specifying a command that didn't do what exactly you wanted. In particular, be careful with analytic functions, which can be immensely finicky with the window specified.

4.3 Data Studio

4.3.1 I can't get my graph to work!

Make sure you're selecting the right parameters. If you don't select "line interpolation" for example, the graph will jump back to the 0 line as it tries to fill in data it doesn't have. Our data is not granular enough to have these data points.

4.4 I have an issue not listed here!

Just let us know! We'll try and help.

5 Next Steps

The Quandl API can grab up to date historical Most sites with historical pulls do not use "real-time data", but it seems viable to automatically schedule jobs to pull daily data to update our Cloud Storage Bucket.