Build a Sample Stock Trade Database

The following URL points to the Github associated with this lab. If you have git installed, you can change to a directory of your choice and clone the github with this command:

git clone https://github.com/mlochbihler/memsql_build_stockdb

```
Run:
```

```
cd memsql_build_stockdb
```

```
Then:
```

```
Using the SingleStore Client to login to your cluster.
```

Step 1: Create the database

Run:

source create_stock_db.ddl;

Step 2: Load company profile data from S3

Load companylist.csv into the **company** table by using the CREATE PIPELINE command. This company file is found in the public S3 bucket and folder: lochbihler/stockdb.

Run:

source create_company_pipeline.ddl;

Then Run:

source start_company_pipeline.dml;

Step 3: Create data generator functionality

The following are several functions used to generate data. The utility function, marketcap_to_DECIMAL, is used by the seed_trades stored procedure to convert a string value for the market capitalization of a stock into a decimal value.

Run:

source create_markcap_to_decimal_funct.dml;

The seed_trades stored procedure generates rows of “seed” trade data based on an integer input value. This provides a query to rank the companies by market capitalization limited to the top 200.

The stored procedure generates approximately the same number of trades for each company.
Run:

source create_seed_trades.dml;

This last stored procedure, iter_stocks, generates additional trade events using a “random walk” to determine the next price for each company. It will store the current price for each company, then add a random number between -1 and 1 to set the new price for each iteration.

Each iteration inserts new records equal to num_shares. When iter_stocks completes, the total number of records in the trade table will be equal to num_shares * iterations + num_shares.

Run:

source create_iter_stocks.dml;

Step 4: Generate trade data

Now that you have created the functionality to generate stock ticker data, run the following commands to generate the data. Note, this process may take several minutes depending on your cluster topology.

Be patient.. Seed_trades will take approximately 14 seconds to run and iter_stocks should run in less than 35 seconds. When this completes, you will have 1,782,903 rows in your trades table.

Inspect the “Log” and write down how long it took to seed trades and generate stock data?

Run:

source run_seed_trades.dml;

Then Run:

source run_iter_stocks.dml;

Check that you have rows in your trade table. Run:
source check_count_trade.sql;

Step 5: Run analytic queries

Run the following analytical queries (even while iter_stocks inserts records in the background if youprefer). Tryeachquerymorethanonceandnotequeryruntimes.

Query 1

Finds the most traded stocks. This query is very fast over very large volumes of data. Run:
source q1_find_most_traded_stocks.sql;

Query 2

Finds the most “volatile” stocks (highest variance in prices). Run:
source q2_find_most_volatile_stocks.sql;

Query 3

Finds the most “volatile” stocks (highest variance in prices) in the last 5 seconds.

Run:

source q3_find_most_volatile_stocks_last5sec.sql;

Query 4

This is a portfolio aggregation query that uses Common Table Expression (CTE), JOIN, and window functions. It also computes min, max, standard deviation, weighted average, and percentiles for each company stock.

Run:

Source q4_portfolio_aggregation.sql;