Project 3 - Streaming Finance Data with AWS Lambda

For this project, you are tasked with provisioning a Lambda function to generate near real-time finance data records for interactive querying. You're going to basically generate a real-time data pipeline for finance data records for interactive querying.

[Requirements 2](#_Toc133325873)

[Assignment 2](#_Toc133325874)

[Infrastructure 2](#_Toc133325875)

[Data Transformation 3](#_Toc133325876)

[Data Analysis 4](#_Toc133325877)

[Data Visualization 4](#_Toc133325878)

[Artifacts 5](#_Toc133325879)

[Lambda Source Code for DataCollector 5](#_Toc133325880)

[S3 Content from DataCollector 5](#_Toc133325881)

[CSV Output From Query in DataAnalyzer 6](#_Toc133325882)

[Jupyter Notebook for DataVisualization 7](#_Toc133325883)

[README 7](#_Toc133325884)

[Rubric 8](#_Toc133325885)

# 

# 

# Requirements

This project leads you through the process of consuming “real-time” data, processing the data and then dumping it in a manner that facilitates querying and further analysis, either in real time or near real time capacity.

In doing so, you will familiarize yourself with a process that you can leverage in your professional or personal endeavors that require consumption of data that is “always on” and changing very quickly, in sub-hour (and typically) sub-minute intervals.

This project is due **May 21st, 2023, at 11:59PM**.

# Assignment

This assignment is broken into three parts:

* Infrastructure
* Data collection
* Data analysis
* Data visualization

## Infrastructure

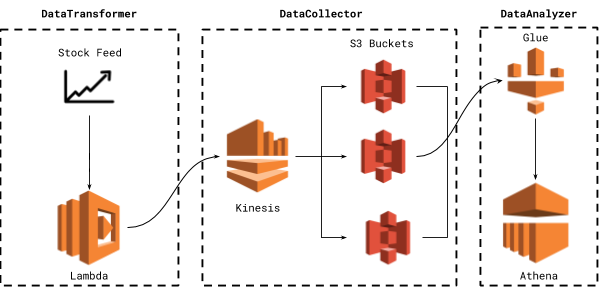
This project consists of four major infrastructure elements that work in tandem:

1. A lambda function that gathers our data (**DataTransformer**)
2. A Kinesis stream that holds our data (**DataCollector**)
3. A serverless process that allows us to query our S3 data (**DataAnalyzer**)
4. Visualizations of the query results (**DataVisualization**)

First, you will create a **Kinesis Delivery Stream** as we did in Lecture 11. Then, you will write a **Lambda function** as we discussed in Lecture 12 to collect the data and push it to the Kinesis Delivery Stream. It will grab stock price data and place it into the delivery defined in the **DataTransformer**.

Then, configure **AWS Glue**, pointing it to the **S3 Bucket** you created in your **DataCollector.** This will allow us to now interactively query the **S3** files generated by the **DataTransformer** using **AWS Athena** to gain insight into our streamed data. This is what I am calling the **DataAnalyzer**. I will talk about this in Lecture 13.

Finally, you will take your query results and generate two visualizations. You will use a **Jupyter Notebook** to generate the visualizations. This is what I am calling **DataVisualization.**



As you can see from this diagram, on the left side we have a Lambda function that can programmatically pull data in from some internet source (more on that in the next section). On the right side we have AWS Athena, which allows us to write ad-hoc, interactive queries against the data we have accumulated to gain various types of insights.

## Data Transformation

In our collector lambda, using the [yfinance](https://pypi.org/project/yfinance/) module ([documentation](https://github.com/ranaroussi/yfinance) here), you will grab pricing information for each of the following e-commerce stocks:

* Amazon (AMZN)
* Alibaba Group (BABA)
* Walmart (WMT)
* eBay (EBAY)
* Shopify (SHOP)
* Target (TGT)
* Best Buy (BBY)
* The Home Depot (HD)
* Costco (COST)
* Kroger (KR)

[yfinance](https://pypi.org/project/yfinance/) is a Python library which allows you to connect to and read stock data. You have to explore how this library works.

You are tasked with collecting **ten full day’s worth of stock HIGH and LOW prices** for each company listed above **between Apr 10th, 2023, and Apr 21th, 2023**, at a **five**-**minute interval**. Note that by “full day” we mean one day of stock trading, which **is not 24 hours**, and the stock market is not open during the weekend.[[1]](#footnote-1) You can use either ‘[history](https://algotrading101.com/learn/yfinance-guide/)’ function or ‘[download](https://pypi.org/project/yfinance/)’ function from yfinance library to get data historical data.

For each datapoint, you will generate a JSON object that looks like so:

|  |
| --- |
| {  "high": 67.5,  "low": 64.61,  "volatility": 2.89,  "ts": "2020-05-13 09:30:00-04:00",  "name": "AMZN"  } |
|  |

This is an example of a single “record” that you would place into the kinesis stream defined in DataCollector. You will have different high, low, and volatility values per hour per company.

Notes:

1. volatility: It refers to the difference between high and low values. You will need to calculate this value per record.
2. ts: You will need to convert a DateTime object to its equivalent string.

You will put your algorithm into sleep for 5 second after you send one data point to Kinesis Stream.

## Data Analysis

We want to prep this data gathered for analysis! To do so, set up a Glue crawler so that you can run AWS Athena queries against your data. Then, in Athena, write and run a query that gives us **the average volatility, the highest volatility, and the lowest volatility per company per day from the list above. [[2]](#footnote-2)**

## Data Visualization

You will take query results and generate the two visualizations on the data you accumulated.

1. Graph the maximum volatility trend per company (A single Line Chart: Each line refers to a company)

Which company is the most volatile?

1. Graph the daily average volatility per company (A Grouped Bar Chart: Each group refers to a company and the bars refer to the daily highest volatility)

Do the findings from this graph support your conclusion from the first graph?

If you select to generate extra two visuals, you will get 1 point for each visual. Feel free to construct new measures using the data you have. For instance, you can normalize your data and use the normalized data in your visuals. You can also run a different query in Athena and generate new results to be used for the extra credit.

**Requirements:**

1. Do not use a line chart or bar chart. Use difference charts. If you select to generate extra two visuals, each should be different from each other as well (*i.e. do not use a pie chart twice*).
2. Ask a question that could be answered by the visuals. You should indicate your answer as well.

# Artifacts

You are to create and submit a zip file. When I unzip your artifact, I expect to see a **single “project03”** folder which contains the following structure:

**project03**

**+-- Athena**

**+-- +-- results.csv**

**+-- +-- query.txt**

**+-- data\_transformer.py**

**+-- Jupyter Notebook**

**+-- +-- Analysis.ipynb**

**+-- +-- Analysis.pdf**

**+-- assets**

**+-- +-- kinesis\_monitor.jpeg**

**+-- +-- screenshot\_of\_s3\_bucket.jpeg**

**+-- +-- exec\_results.jpeg**

**+-- +-- results.jpeg**

**+-- README.md**

## Lambda Source Code for DataCollector

I’d like the source code for your lambda function that collects the data from yfinance and puts it into the firehose stream. You **must** name your lambda function file **data\_transformer.py** and place it in your project root folder.

You will create environmental variables for region parameter in **boto3.client()** function and StreamName parameter in **put\_record()** function. **Take a look at the extra credit assignment to see how to create an environmental variable in Lambda.**

## S3 Content from DataCollector

**screenshot\_of\_s3\_bucket.png**

Take a full-page screenshot of your S3 bucket that shows the file structure (i.e. 2022/04/27/etc) and some S3 flat files in that folder. See the example below:

Graphical user interface, text, application

Description automatically generated

## CSV Output From Query in DataAnalyzer

Please upload a file called **results.csv** that contains your query output. Expected columns: *company, date, average volatility, highest volatility, lowest volatility*

Please also include a file called **query.txt** that contains the actual query that you ran to generate your **results.csv** file and **results.jpeg** that is the screenshot of the results from AWS Athena.

**results.jpeg**

Graphical user interface, application, table

Description automatically generated

## Jupyter Notebook for DataVisualization

You will take your **results.csv** file and generate two visualizations using a **Jupyter Notebook**. This file must be called **Analysis.ipynb** and **Analysis.pdf**. You will submit both files.

## README

The **README**, in markdown or pdf, should contain the following items:

* A brief blurb describing the project and the technology leveraged to conduct your analysis. This ought to be brief and informational, in case folks in the future want to recreate your results.
* Answers to the visualization questions

**kinesis\_monitor.jpeg**

Provide a screenshot of your AWS Kinesis configuration page. See the example below:

A screenshot of a computer

Description automatically generated

**exec\_results.jpeg**

A screenshot of a computer

Description automatically generated

# Rubric

|  |  |
| --- | --- |
| **RUBRIC** | |
| **Structure & Naming** | **8** |
| Parent folder is named “project03” and is exposed correctly when unzipped | 1 |
| **README** is available directly in the project03 root folder (1 pt). It contains a brief blurb about the project (1pt). It contains the answers to the visualizations (2pts). | 4 |
| “assets” folder exists | 1 |
| “Athena” folder exists | 1 |
| “Jupyter Notebook” folder exists | 1 |
| **DataTransformer** | **11** |
| Lambda function source code imports **yfinance** module and correctly retrieves ten full day’s worth of stock data from companies listed on the given time frame | 3 |
| Lambda function source code correctly transforms the dataframe from **yfinance** module into JSON in the correct format (specified in this document). | 3 |
| Lambdafunction calls **boto3.client(kinesis)** to put records successfully (with newline!) into the kinesis delivery stream. | 2 |
| Algorithm is put into sleep for 5 second after one data point is sent to Kinesis Stream. | 1 |
| An environment variable is used for region parameter in **boto3.client()** function. | 1 |
| An environment variable is used for StreamName parameter in **put\_record()** function. | 1 |
| **DataCollector** | **6** |
| Data is collected via Lambda Function (proven through **exec\_results.jpeg**). The screenshot is available in the assets folder. | 2 |
| **Kinesis** stream is configured properly and drops data to firehose delivery stream (proven through **screenshot\_of\_s3\_bucket.jpeg**). The screenshot is available in the assets folder. | 2 |
| **Kinesis** stream was utilized in generating the data (proven through **kinesis\_monitor.jpeg).** The screenshot is available in the assets folder. | 2 |
| **DataAnalyzer** | **9** |
| **results.csv** and **query.txt** files are available in the Athena folder. | 2 |
| The query is run through AWS Athena (proven through **results.jpeg**). The screenshot is available in the assets folder. | 2 |
| **results.csv** contains data that correctly identifies the daily average volatility, the daily highest volatility, and the daily lowest volatility. Expected columns: *company, date, average volatility, highest volatility, lowest volatility* | 2 |
| **query.txt** file contains a query (with the correct syntax) that can generate the output in results.csv if run. | 3 |
| **DataVisualization** | **6** |
| **Analysis.ipynb** and **Analysis.pdf** are available in Jupyter Notebook folder. | 2 |
| **TWO** visuals are correctly created | 4 |

# 

1. You will collect a total of 7800 data points. [↑](#footnote-ref-1)
2. This query will give you a total of 100 data points. [↑](#footnote-ref-2)