



MTech EBAC AY2019/2020

Big Data Engineering & Web Analytics Practice Module

Stock Market Returns Correlation Analysis

TEAM MEMBER

- | |
|-----------------------------|
| 1. EE MENG HOCK |
| 2. MRITUNJAY KUMAR |
| 3. SUM KWOK HOONG |
| 4. TAN DEHONG |
| 5. TAN WEE HAN COLIN |

A. INTRODUCTION

Financial markets trade round the clock globally. Our analysis seeks to explore the degree of correlations between stock price movements for selected stocks in the same and different regions.

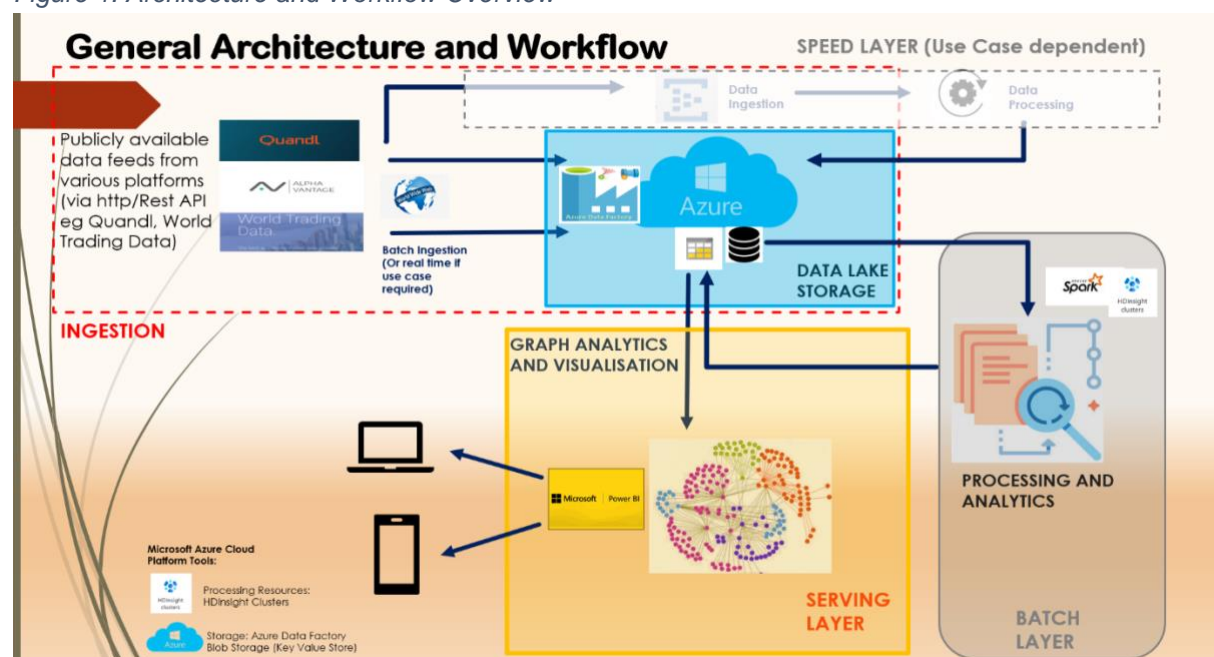
Understanding such correlations would be useful for professional fund managers and investment firms holding diversified portfolios. These results could be used for short- to mid-term trading opportunities or portfolio positioning purposes.

With the advent of high-frequency trading, vast volumes of trading data could be generated each day, with observations within a single day equivalent to 30 years of historical daily data.

A big scalable data architectural framework is needed in order to process such volumes of data in real-life usage. For this project, we pull daily price movement data from publicly available sources, ingest and transform the data for correlation analysis and provide output with the data for graph analytics and visualisation as shown in Figure 1.

Azure Cloud platform was used to ingest, process, analyse and store the data. We have chosen Azure Cloud primarily for its rate of innovation (new features and functionalities are continuously being added) and quality integration with a one-stop shop of other compatible tools (for ingestion, data queries and visualisation) needed for this project.

Figure 1: Architecture and Workflow Overview



B. DATA SOURCES

Yahoo Finance API has long been a reliable data source for many investors. However, Yahoo decided to shut down the Yahoo Finance API in May 2007. There are some alternative sources, of which we explored the following three.

1. *Alpha Vantage* (Website: <https://www.alphavantage.co/>)

Alpha Vantage Inc. is a leading provider of various free APIs. Making use of the API enables anyone to gain access to historical and real-time stock data, FX-data, and cryptocurrency data. Extensive documentation on the APIs can be found at the website.

With Alpha Vantage, a person can perform up to 5 API-requests per minute at no charge.

The maximum requests per day are limited to 500.

2. *World Trading Data* (Website: <https://www.worldtradingdata.com/>)

World Trading Data provides RESTful API endpoints. They return the market data in near-time. Also, historical data is available.

World Trading Data does provide free access to their API. There are different plans available for data access. The free plan covers up to 5 stocks per request in the real-time API, and full intraday data API and currency API access are also given.

3. *Quandl* (Website: <https://www.quandl.com/>)

Quandl is a marketplace for data in the state of the art formats, like Python, Excel, Matlab, R and more. The website is impressively well designed, and the outlined processes make sense.

However, for a new free account at [quandl.com](https://www.quandl.com/), one can gain access to only free data samples and some useful historical data for various but less exciting data sets.

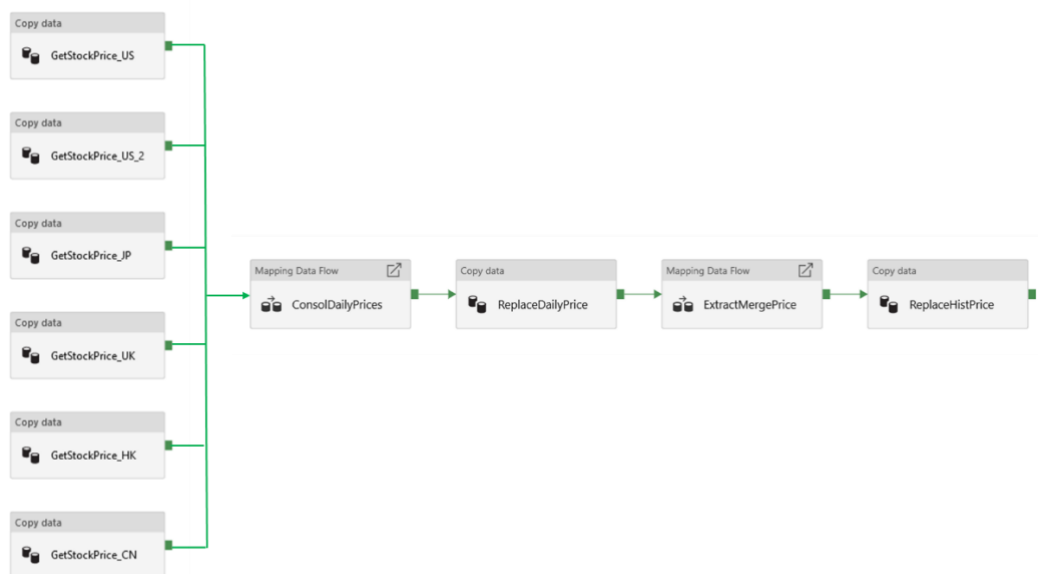
World Trading Data is the primary data source for the data used in this project.

C. DATA INGESTION

Azure Data Factory is a cloud-based ETL and data integration service that allows one to create data-driven workflows for orchestrating data movement and transforming data at scale.

We used the Azure Data Factory (ADF) to create and schedule data-driven workflows (called pipelines) to ingest data from disparate data sources in this project. We built an ETL process that first connects to data sources via API and collects the daily stock closing prices of 30 counters for 5 countries (namely United States, United Kingdom, Japan, Hong Kong and China) every morning. The ingested data are then consolidated, transformed and appended or saved the required data into Azure Blob storage using ADF mapping data flows as shown in Figure 2.

Figure 2: Ingest, prepare and transform using Azure Data Factory



Connect and Collect

The Copy Activity in ADF was used to copy data from a REST endpoint by:

1. Creating Linked Services for the source data store and the sink datastore

Linked services are much like connection strings, which define the connection information needed for Data Factory to connect to external resources. Click on the “Connections” at the Factory Resources pane in ADF to add a link service. Provide a name to the Link Service, the Base URL and Authentication type, as

shown in the diagram below. We can perform a Test Connection to verify the configuration.

We created a total of 6 link services to copy 30 stock data as we are limited by the 5 stocks per request for no-charge access to the data source.

Figure 3: Input parameters for importing source data via HTTP

Edit linked service (HTTP)

Name *
HttpServer_USStocks

Description

Connect via integration runtime *
AutoResolveIntegrationRuntime

Base URL *
https://api.worldtradingdata.com/api/v1/stock?symbol=AAPL,AMD,CSCO,INTC,MSFT&output=csv&sort=

Server Certificate Validation
☒ Enable ☐ Disable

Authentication type *
Anonymous

Annotations
+ New

► Advanced ⓘ

Apply Test connection Cancel

2. Creating Datasets for the source and sink

Next, click on the “Action” button beside the “Datasets” in the Factory Resource pane to add a dataset. The dataset represents the structure of the data within the linked data stores. To create a dataset, we need to define the connection and schema of the source data, as shown below.

We created 6 Http-source datasets, one for each link service configured.

Figure 4: Fixing the schema for imported source data

The screenshot shows the 'Import schema' dialog box in Azure Data Factory. The 'General' tab is selected, displaying the 'Name *' field with the value 'HTTP_Source_US1' and an empty 'Description' field. Below this, the 'Connection' tab is visible, showing 'Linked service *' set to 'HttpServer_USStocks'. The 'Schema' tab is active, showing a table of columns with names and types. The 'Import schema' button is highlighted.

Column name	Type
Symbol	String
Name	String
Currency	String
Price	String
Price Open	String
Day High	String
Day Low	String
52 Week High	String
52 Week Low	String
Day Change	String
Change %	String
Close Yesterday	String
Market Cap	String

3. Creating a pipeline with the Copy activity

A pipeline run in Azure Data Factory defines an instance of pipeline execution. For example, say you have a pipeline that executes at 8:00 AM, 9:00 AM, and 10:00 AM. In this case, there are three separate runs of the pipeline, or pipeline runs.

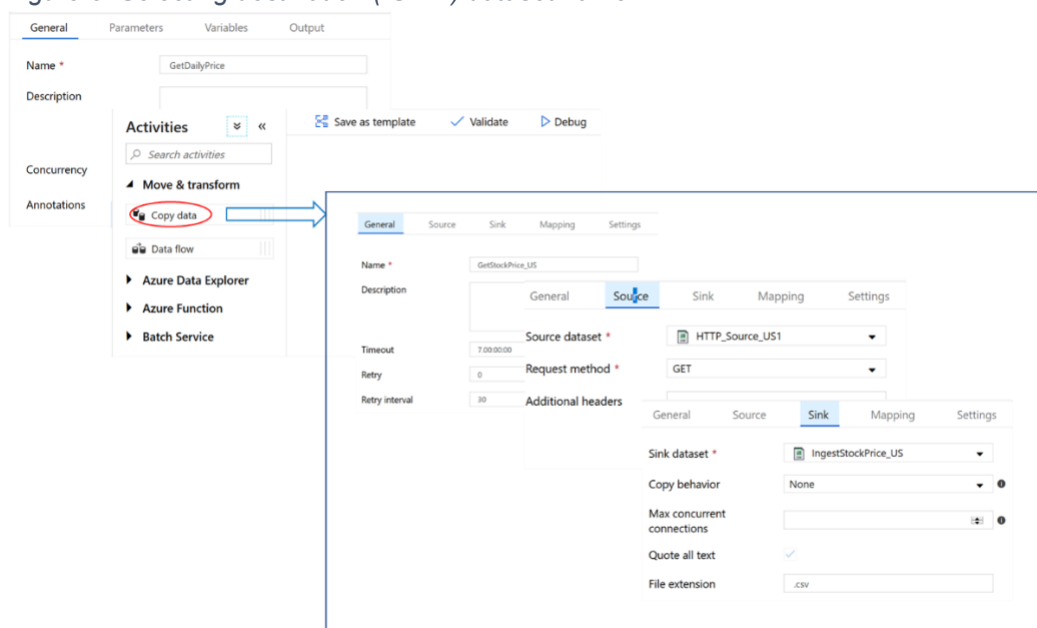
For the project, we created a basic pipeline named GetDailyPrice with 6 Copy Data activities to get the stock prices from the public data source to a destination folder in the Azure Blob storage. We created 6 more datasets to save the data received from the source datasets.

To create a pipeline, click on the “Action” button beside the “Pipelines” in the Factory Resource pane. Expand the “Move & Transform” in the “Activities” pane and drag & drop the “Copy Data” option to the right pane. Name the Copy Data activity and configure the Source dataset and Sink dataset as shown in the diagram below.

We created 6 Copy Data activities, one for each link service/dataset configured, as shown in Figure 2 above.

We also configured a Trigger to execute the GetDailyPrice pipeline at 6 am Singapore time every morning to fetch the closing prices of all selected stocks.

Figure 5: Selecting destination (“Sink”) dataset name



Below is a sample of the ingested data saved in Azure Blob storage.

Figure 6: Sample of original U.S. data imported from a source

Symbol	Name	Currency	Price	Price Open	Day High	Day Low	52 Week High	52 Week Low	Day Change	Change %	Close Yesterday	Market Cap	Volume	Volume Avg	Shares	Stock Exchange Long	Stock Exchange Short	Timezone	Timezone Name	GMT Offset	Last Trade Time	PE	EPS
AAPL	Apple Inc.	USD	263.19	265.54	266.08	260.40	268.00	142.00	-3.10	-1.16	266.29	1189408145408	24267717	22293000	4519199744	NASDAQ Stock Exchange	NASDAQ	EST	America/New_York	-18000	2019-11-20 16:00:01	22.14	11.89
AMD	Advanced Micro Devices, Inc.	USD	40.98	40.96	41.75	40.07	41.79	16.03	-0.31	-0.75	41.29	46659825664	78143621	65583957	1138599936	NASDAQ Stock Exchange	NASDAQ	EST	America/New_York	-18000	2019-11-20 16:00:01	214.55	0.19
CSCO	Cisco Systems, Inc.	USD	45.08	45.69	45.66	44.93	58.26	40.25	-0.39	-0.86	45.47	191311872000	17510504	28674228	4243830016	NASDAQ Stock Exchange	NASDAQ	EST	America/New_York	-18000	2019-11-20 16:00:01	17.94	2.51
INTC	Intel Corporation	USD	57.90	58.26	58.36	57.37	59.59	42.86	-0.45	-0.77	58.35	251865006080	16974724	14272085	4350000128	NASDAQ Stock Exchange	NASDAQ	EST	America/New_York	-18000	2019-11-20 16:00:01	13.55	4.27
MSFT	Microsoft Corporation	USD	149.62	150.31	150.84	148.46	151.33	93.96	-0.26	-0.17	149.88	1141422555136	23118304	19801228	7695349760	NASDAQ Stock Exchange	NASDAQ	EST	America/New_York	-18000	2019-11-20 16:00:01	28.23	5.30

Consolidate and Transform

To consolidate and transform the 6 datasets received from the data sources, we created another pipeline “Daily_UpdateHistPrice” with Data Flow and Copy Data activities, as shown below. Mapping data flow in ADF allows us to develop graphical data transformation logic without writing code. ADF handles all the code translation, path optimisation, and execution of the data flow jobs on the scaled-out cluster.

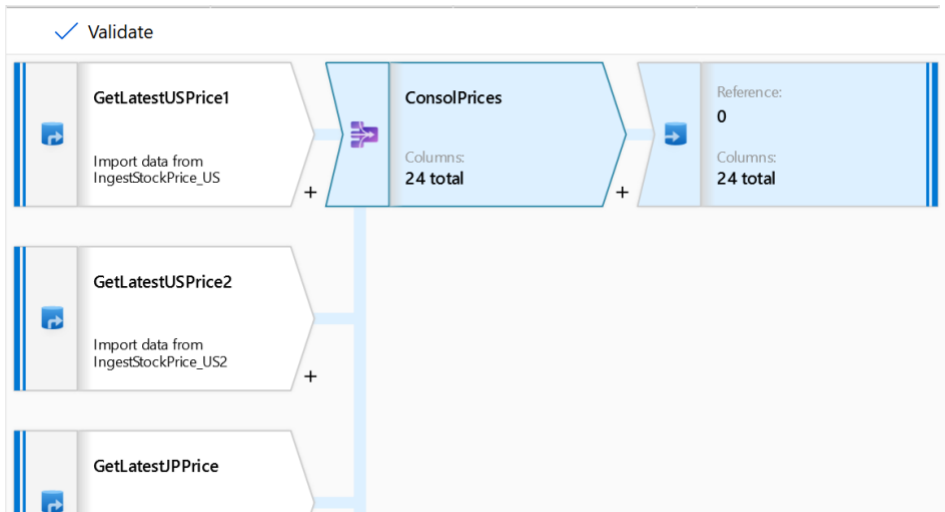
Figure 7: Overall process flow to consolidate daily prices and to merge with historical data



1. Consolidate Daily Prices


To create a data flow, select the plus sign under Factory Resources, and then select Data Flow followed by Mapping Data Flow. The Data Flow takes you to the canvas where you can create the transformation logic, as shown below. Select Add source to start configuring your source transformation.

Figure 8: Data flow in Data Factory to consolidate daily prices



We need first to configure the “source” to get all 6 datasets saved from the source data. Name the source and define the source dataset as shown below, and repeat the step for all 6 source datasets.

Figure 9: Creating destination file names for various countries' daily updated prices

Source Settings	Source Options	Projection	Optimize
Output stream name *	<input type="text" value="GetLatestUSPrice1"/>		
Source dataset *	 IngestStockPrice_US ▼		
Options	<input checked="" type="checkbox"/> Allow schema drift ⓘ <input type="checkbox"/> Infer drifted column types ⓘ <input type="checkbox"/> Validate schema ⓘ <input type="checkbox"/> Multiline rows ⓘ		
Skip line count	<input type="text" value=""/>		
Sampling *	<input type="radio"/> Enable <input checked="" type="radio"/> Disable ⓘ		



Then click on the plus sign on the lower right of one of the source icon to add a new “union” transformation. Enter the names of the other 5 source streams to the Union Settings, as shown below.

Figure 10: Selecting the streams of daily prices to consolidate under a single list

Union Settings	Optimize	Inspect	Data Preview
Output stream name *	<input type="text" value="ConsolPrices"/>		
Incoming stream *	GetLatestUSPrice1		
Union by *	<input checked="" type="radio"/> Name <input type="radio"/> Position ⓘ		
Union with	Streams <input type="text" value="GetLatestUSPrice2"/> <input type="text" value="GetLatestJPPrice"/> <input type="text" value="GetLatestCNPrice"/> <input type="text" value="GetLatestUKPrice"/> <input type="text" value="GetLatestHKPrice"/>		

To save the output, click on the plus sign on the lower right of the “ConsolPrices” icon to add a “sink”. Click on “New” to create a new dataset to keep the output.

Figure 11: Setting the destination file name for consolidated daily prices

Sink	Settings	Mapping	Optimize	Inspect	Data Preview
Output stream name *	<input type="text" value="PutConsolPrice"/>	Documentation			
Incoming stream *	ConsolPrices				
Sink dataset *	 ConsolDailyPriceSink ▼	 Edit	+ New		
Skip line count	<input type="text" value=""/>				
Options	<input checked="" type="checkbox"/> Allow schema drift ⓘ <input type="checkbox"/> Validate schema ⓘ				

2. *Replace Daily Prices dataset*

As we encountered a problem using the “CosolDailyPriceSink” dataset as-is when the new daily price data came in, we added a Copy Data activity to copy the “CosolDailyPriceSink” into another dataset named “CosolDailyPriceFile”, which can be overwritten daily, as shown below.

Figure 12: Setting the required destination parameters

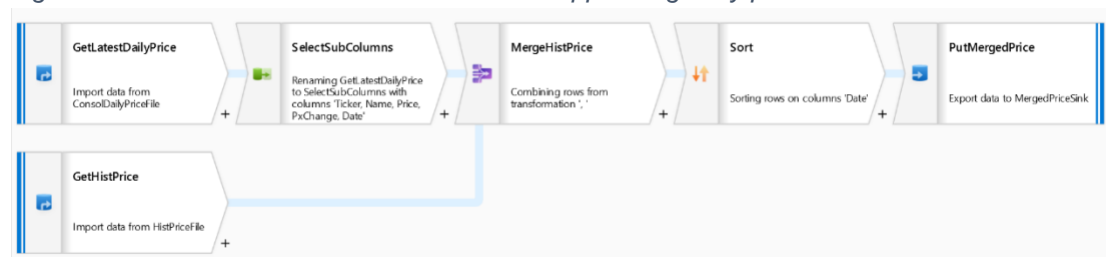
The screenshot shows the configuration for a Copy Data activity named "ReplaceDailyPrice". The "General" tab is active, showing the source dataset as "CosolDailyPriceSink". The "Sink" tab is also visible, showing the destination dataset as "CosolDailyPriceFile". The "Copy behavior" is set to "Merge files", and the "File extension" is ".txt".

Tab	Property	Value
General	Name *	ReplaceDailyPrice
	Description	
	Timeout	7:00:00:00
	Retry	0
Source	Source dataset *	CosolDailyPriceSink
	Recursively	<input checked="" type="checkbox"/>
Sink	Sink dataset *	CosolDailyPriceFile
	Copy behavior	Merge files
Mapping	Max concurrent connections	1
	Quote all text	<input checked="" type="checkbox"/>
Settings	File extension	.txt

3. *Replace Historical Price dataset*

Another Mapping Data Flow “ExtractMergePrice” was created to append the latest daily prices to the historical data collected, as shown below.

Figure 13: Process flow to transform data and appending daily prices to historical data



- “GetLatestDailyPrice” and “GetHistPrice” are source nodes to get the “CosolDailyPriceFile” dataset and “HistPriceFile” dataset respectively.
- “SelectSubColumns” is a “select” schema modifier to rename and select the columns needed for the analysis.
- “MergeHistPrice” is a “union” transformation to append the daily prices to the historical prices.
- “Sort” is a “row” modifier to sort the dataset by ‘Date’.

- e. “PutMergePrice” is a “sink” to save the output to “MergedPriceSink” dataset.

4. Consolidate Daily Prices

Similar to step 2, we added a Copy Data activity to copy the “MergedPriceSink” dataset into another dataset named “HistPriceFile” dataset.

The resulted dataset is in the format below, ready for further processing and analysis.

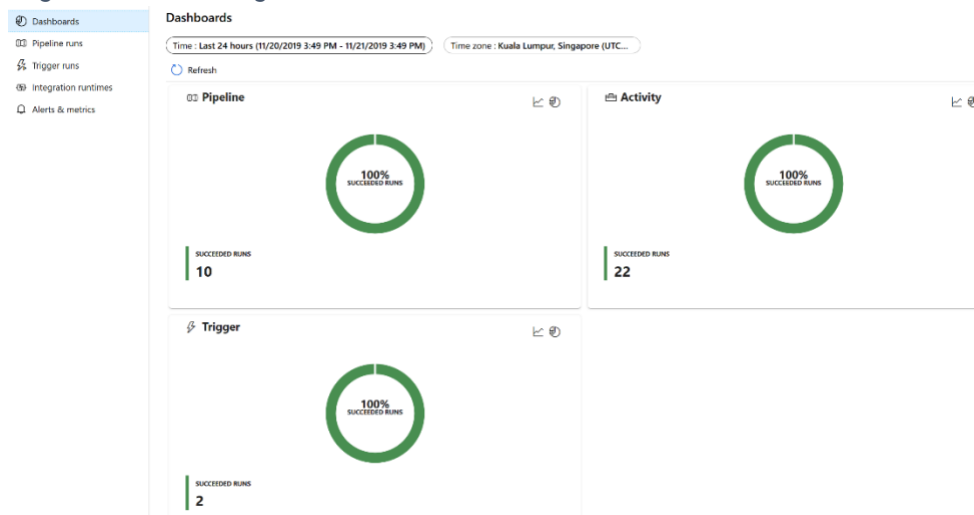
Figure 14: Sample of a merged data file (historical plus new daily price)

Ticker	Name	Price	PxChange	Date
CMCSA	Comcast Corporation	44.22	-1.33	2019-11-20
FB	Facebook, Inc.	197.51	-0.91	2019-11-20
MU	Micron Technology, Inc.	45.57	-2.17	2019-11-20
SIRI	Sirius XM Holdings Inc.	6.9	-1.0	2019-11-20
ZNGA	Zynga Inc.	6.23	-1.11	2019-11-20
CNAL	Centrica PLC	72.7	-1.17	2019-11-20
MNG.L	M&G PLC	232.8	-0.26	2019-11-20
RDSA.L	Royal Dutch Shell Plc	2270.0	-0.7	2019-11-20

Monitoring Pipelines

Once the pipelines have been created and published in Azure Data Factory, we can monitor the pipeline runs natively in the Azure Data Factory user experience. To open the monitoring experience, click on the Monitor icon on the left sidebar.

Figure 15: Monitoring dashboard



D. DATA PROCESSING & ANALYTICS

Azure HDInsight is a cloud distribution of Hadoop components. We signed up at Azure Portal for a free account with 12 months of free service to use HDInsight for the project. Azure HDInsight provides popular open-source frameworks such as Hadoop, Spark, Hive, LLAP, Kafka, Storm, R, and many others.

HDInsight includes specific cluster types such as Apache Hadoop, Apache Spark, Apache HBase, Apache Storm, etc. We selected the Apache Spark cluster, for its parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications. HDInsight makes it easier to create and configure a Spark cluster that is compatible with Azure Storage and Azure Data Lake Storage.

Create HDInsight-Spark Cluster

1. *Create a Resource Group*

At the Azure Portal homepage, click on 'Resource Group' followed by 'Add' to create a resource group. Enter the subscription (free or pay-as-use) and the resource group name. You may need to try a different Region as we encountered some issues with Regions such as South-East-Asia and Australia-South-East.

Click on 'Review + Create' to create the resource group.

Figure 16: Menu to create an HDInsight Cluster

Microsoft Azure Search resources, services, and docs (G+)

Azure services

- Create a resource
- HDInsight clusters
- Resource groups
- Subscriptions
- Azure Cosmos DB
- Storage accounts

Home > Resource groups > Create a resource group

Create a resource group

Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

Project details

Subscription * ⓘ MH_Trial

Resource group * ⓘ BEAD-Project

Resource details

Region * ⓘ (US) Central US

2. Create an HDInsight Cluster

Go back to Azure Portal homepage, click on 'HDInsight clusters' followed by 'Add' to create an HDInsight cluster.

At the Basic tab, select the resource group name created in the earlier step, name the cluster with the region selected earlier, click on Cluster type to select Spark cluster and the version, enter the passwords for cluster login, then click on 'Next: Storage'.

Figure 17: More settings for Cluster creation

Create HDInsight cluster

[Go to classic create experience](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

MH_Trial

Resource group *

BigData

[Create new](#)

Cluster details

Name your cluster, pick a region, and choose a cluster type and version. [Learn more](#)

Cluster name *

BigData2

Region *

East US

Cluster type *

Spark

[Change](#)

Version *

Spark 2.3 (HDI 3.6)

Cluster credentials

Enter new credentials that will be used to administer or access the cluster.

Cluster login username * ⓘ

admin

Cluster login password *

.....

Confirm cluster login password *

.....

Secure Shell (SSH) username * ⓘ

sshuser

Use cluster login password for SSH

☒

Review + create

« Previous

Next: Storage »

3. Create Azure Storage

At the Storage tab, select the Primary storage type, either Azure Storage (Blob) or Azure Data Lake Storage Gen 1/2. Name the Primary storage account and container, all in lower case alphanumeric characters, then click on the 'Next: Security + networking' button at the bottom of the page.

We will leave the 'Security + networking' tab as is and move on to the 'Configuration + pricing' tab.

Figure 18: Settings for creating Azure Blob Storage

The screenshot shows the 'Storage' tab in the Azure portal. At the top, there are five tabs: 'Basics', 'Storage' (which is highlighted with a purple underline), 'Security + networking', 'Configuration + pricing', and 'Review + create'. Below the tabs, there is a heading 'Primary storage' followed by a description: 'Select or create a storage account that will be the default location for cluster logs and other output.' Below this, there are three main configuration sections. The first section is 'Primary storage type *', which has a dropdown menu set to 'Azure Storage'. The second section is 'Selection method * ⓘ', which has two radio buttons: 'Select from list' (which is selected) and 'Use access key'. The third section is 'Primary storage account *', which has a dropdown menu set to '(New) bigdata2hdistorage' and a link 'Create new' below it. Below the 'Primary storage account' section, there is a 'Container * ⓘ' section with a text input field set to 'default' and a green checkmark icon to its right. Below the 'Primary storage' section, there is a heading 'Data Lake Storage Gen1' followed by a description: 'Provide details for the cluster to access Data Lake Storage Gen1. The cluster will be able to access any Data Lake Storage Gen1 accounts that the chosen service principal has access to.'

Basics **Storage** Security + networking Configuration + pricing Review + create

Select or create storage accounts that will be used for the cluster's logs, job input, and job output. Configure the cluster's access to these accounts, if needed.

Primary storage

Select or create a storage account that will be the default location for cluster logs and other output.

Primary storage type * Azure Storage

Selection method * ⓘ ☒ Select from list ☐ Use access key

Primary storage account * (New) bigdata2hdistorage

Create new

Container * ⓘ default

Data Lake Storage Gen1

Provide details for the cluster to access Data Lake Storage Gen1. The cluster will be able to access any Data Lake Storage Gen1 accounts that the chosen service principal has access to.

4. Configure Cluster Nodes

At the 'Configuration + pricing' tab, select the Node size for the Head node and Worker node. For development or sandbox, you may choose smaller node sizes to save cost. Click on 'Review + create' to verify the cluster configuration.

Figure 19: Configuring the number of nodes and processing power required

[Basics](#)
[Storage](#)
[Security + networking](#)
[Configuration + pricing](#)
[Review + create](#)

Configure cluster performance and pricing. [Learn more](#)

Node configuration

Configure your cluster's size and performance, and view estimated cost information.

The cost estimate represented in the table does not include subscription discounts or costs related to storage, networking, or data transfer.

i This configuration will use 20 of 250 available cores in the East US region.
[View cores usage](#)

Add application

Node type	Node size	Number of ...	Estimated cost/hour
Head node	A2m v2 (2 Cores, 16 GB RAM), 0.15 USD/... <input type="text"/>	2	0.31 USD
Worker node	A2m v2 (2 Cores, 16 GB RAM), 0.15 USD/... <input type="text"/>	8 <input checked="" type="checkbox"/>	1.22 USD

Review + create

« Previous

Next: Review + create »

Once the validation is succeeded, review the configuration again before clicking on the 'Create' button. The cluster will take approximately 20 mins to deploy.

Process and Analyse Data

Correlation Calculation and Creating Spark Data Frames for Nodes and Edges

We used the Jupyter Notebook provided by the HDInsight Spark Cluster to develop the program to process and analyse the stock price data. The merged price data file "HistPriceFile" was read from the Azure Blob Storage to start the data processing. The correlation values between different pairs of stocks are calculated and presented as an adjacency matrix table. These values will eventually be used as the weights for the edges between pairs of stocks on the graphs.

Two Spark data frames were created to store the network graph data, one each for Nodes and Edges. Node Values are made up of Ticker Names from the merged price data file, while values for Edges are the correlation values from the adjacency data frame derived from the correlation matrix during the data processing. These

data frames are then saved into the Blob Storage container. At the same time, we saved them to a tabular format in Azure Table Storage for retrieval by Power BI Visualizer.

See Codes below also included as separate files attached along with this document

PySpark codes: Correlation computation and conversion to graph data in Appendix for code details.

E. DATA VISUALISATION

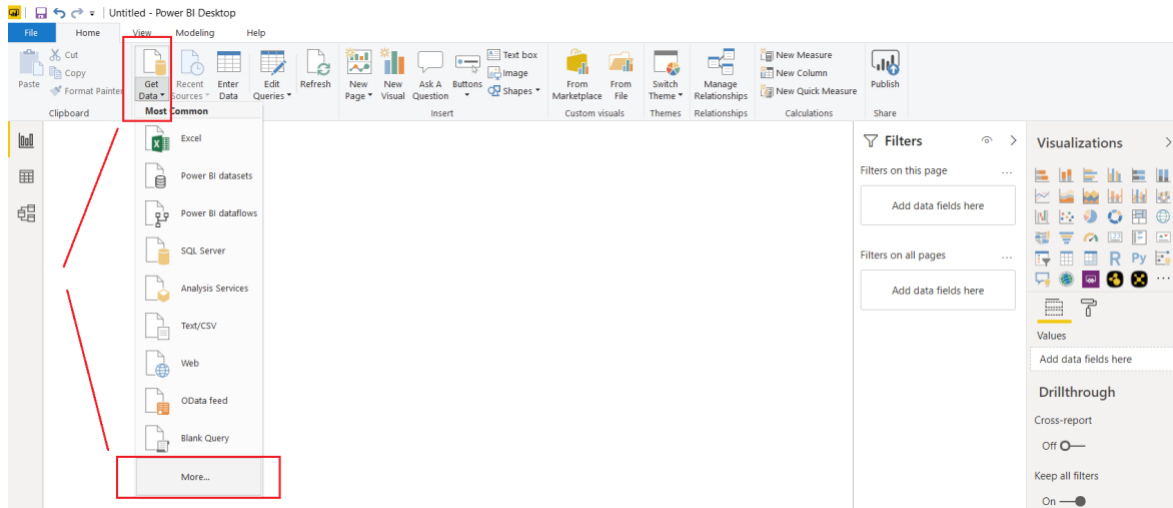
Activating our Visualizer (Power BI) to read in data from Data Source (Azure Table Storage)

After the data ingestion, processing, and analysing, the project objective is not complete without an intuitive medium to disseminate the information and analytics insights to the business users. We require a suitable programme to enable us to plot network graph visualisation and publish the analytics results. There are 2 popular data visualisation programmes (Power BI and Tableau) available to us and being evaluated for their suitability for this project. Power BI is a winner for this project for its support for generating network graph visualisation.

With the full set of Node (Stock Ticker) and Edge (Correlations) information ready for presentation in an actual graphical format, we are ready to read in the data into our graph visualizer. For our case, we have chosen to utilise Microsoft's Power BI (download available at <https://powerbi.microsoft.com/en-us/desktop/>).

Reading in the data source for the graph visualisation is straightforward via the Get Data >> More >> Azure >> Azure Table Storage buttons to extract the node and link data in our Azure Table Storage.

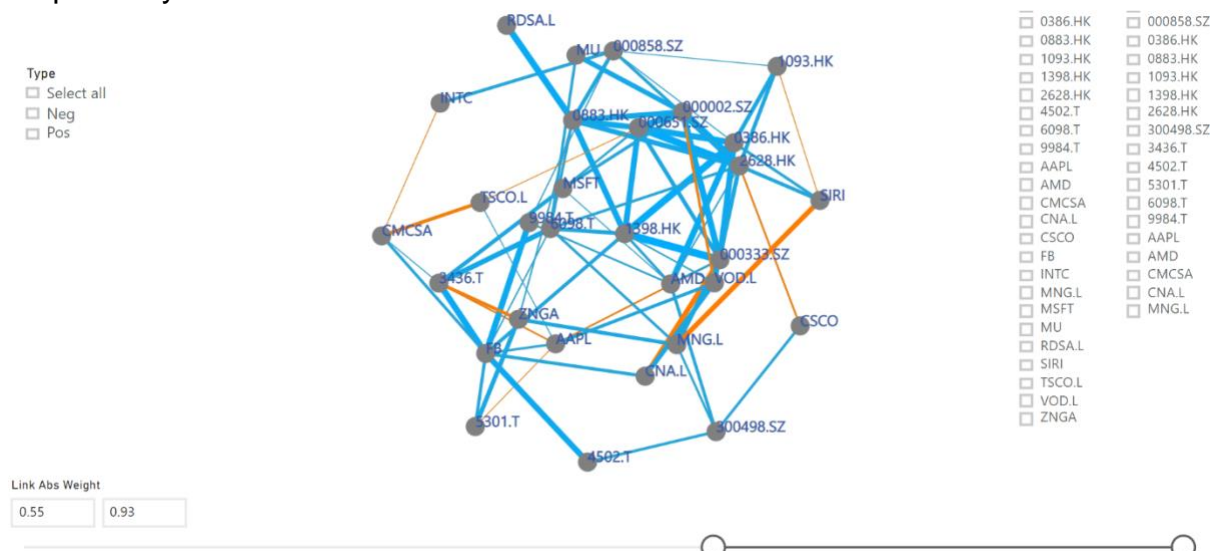
Figure 20: Reading in the Node and Link data from Azure Table Storage



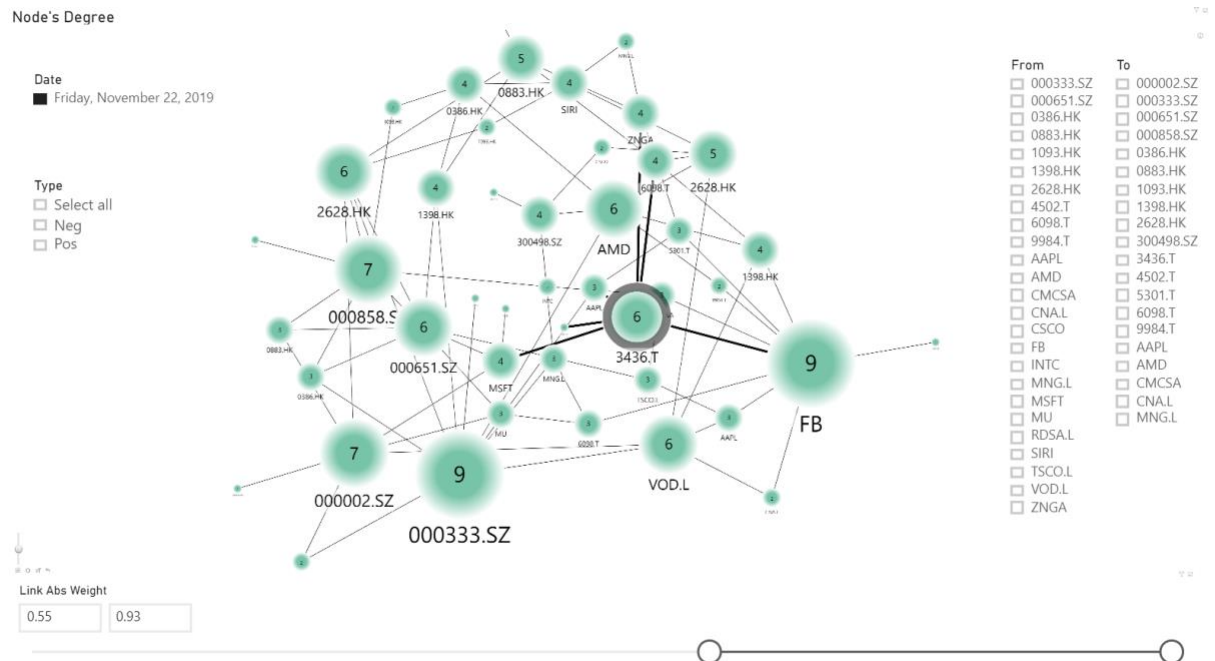
Once the data has been loaded in, we can then make customisations to our graph outputs with variations such as the below. Some of the interactive features include the ability to manually adjust the correlation scale according to user preference, selecting by specific dates and correlation type (positive only, negative only or both).

General Overview of Stocks Network

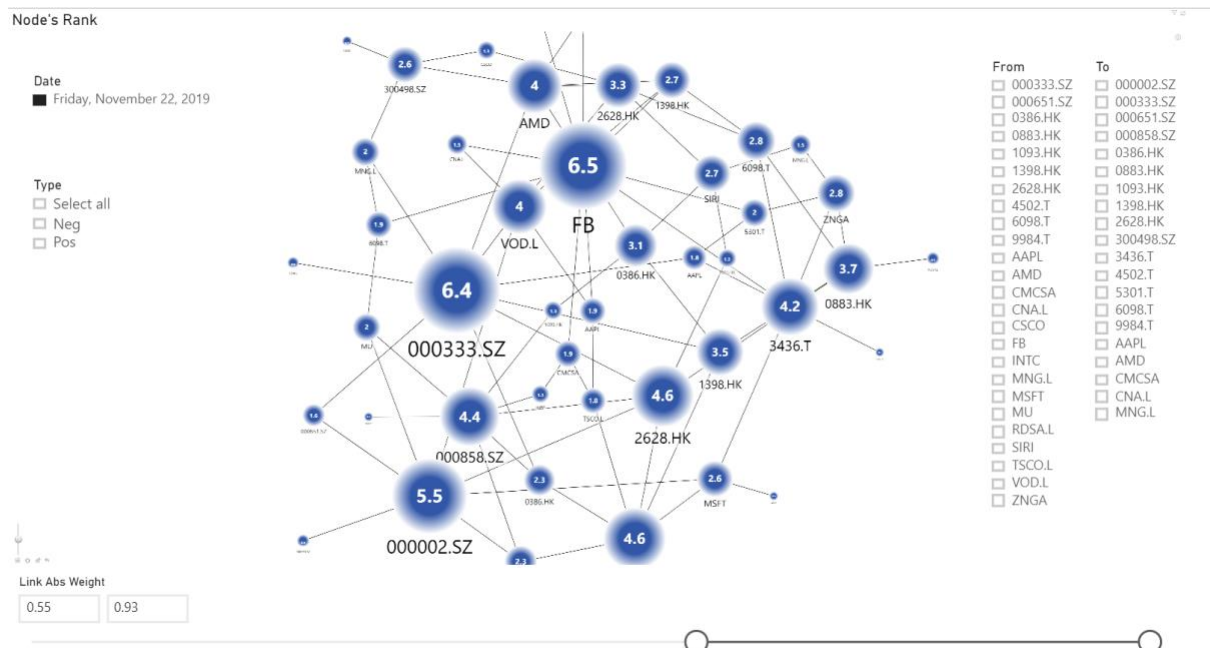
We have incorporated Edge colours and thickness levels to enhance the visualisation, representing the direction of correlation (positive/negative) and strength of correlation respectively (thicker line indicating stronger correlation) respectively.



Stocks Network with Degrees represented as numbers on the nodes



Stocks Network with Nodes' Ranks represented as numbers on the nodes



Sample stock list and list of possible business questions and answers

Date
 Friday, November 22, 2019

Ticker	Ticker Name	Price	Questions	Response
000002.SZ	VANKE-A/Shs A Vtg 1.00	26.75	Find out basket of stocks which are highly correlated to each other across sectors regions	clique 1 [1398.HK_000333.SZ_000651.SZ_0386.HK] clique 2[000002.SZ_000651.SZ_0883.HK_2628.HK]
000333.SZ	Midea Group Co Ltd	56.20		
000651.SZ	Gree Electric Appliances Inc of Zhuhai	57.45		
000858.SZ	Wuliangye Yibin Co., Ltd.	133.66		
0386.HK	China Petroleum & Chemical Corp	4.42	Find out highly influential stocks	000002.SZ
0883.HK	CNOOC Ltd	11.68	Find out most stable stocks (possibly alpha stocks) with least correlation to other stocks	RDSA.L
1093.HK	CSPC Pharmaceutical Group Ltd	19.98		
1398.HK	INDUST & COMMER/Shs H Vtg 1.00	5.66	How connected a region or sector is	0.172043
2628.HK	China Life Insurance Co Ltd	20.15		
300498.SZ	Wens Foodstuff Group Co Ltd	36.07	Stocks which are more representative of broader market(possibly index stocks)	6098.T
3436.T	Sumco Corporation	1,662.00		
4502.T	Takeda Pharmaceutical Company	4,467.00		
5301.T	Tokai Carbon Co., Ltd.	1,036.00		
6098.T	Recruit Holdings Co., Ltd.	3,955.00		
9984.T	SoftBank Group Corp.	4,152.00		
AAPL	Apple Inc.	262.01		
AMD	Advanced Micro Devices, Inc.	39.52		
CMCSA	Comcast Corporation	44.36		
CNAL	Centrica PLC	79.32		
CSCO	Cisco Systems, Inc.	44.84		

On top of the above visualisations, queries on network characteristics and metrics can also be made via RStudio or the GraphFrames package in Spark or any applicable graph database/visualizer.

F. PUBLISHING THE DATA VISUALISATION REPORTS

For the data visualisation reports to be useful to the business users, we publish the above visualisations to the Power BI web server after ensuring the data are accurately plotted. Power BI web server provides a channel for users with the Power BI accounts to access their reports online. For this project, we further publish our data visualisation for public web access by following this [link](#).

G. USE CASE EXAMPLES

Using the tabulated correlations of stocks and presenting these data in graphical forms, there are multiple use cases for such analysis, primarily in the portfolio management industry for portfolio positioning and/or risk hedging purposes. Below are some possible use cases. (Measures indicate the metrics we need to extract which can be via RStudio or the GraphFrames package in PySpark or any applicable graph database/visualizer).

- i) Find out clusters of stocks which trade with high correlations to each other across sectors/regions.
 - **Measure:** Cliques/Maximal cliques
 - **Practical Application:** Good for trading purposes such as to forecast movements of stocks in later time zones using data from earlier markets or to find out stocks highly correlated with portfolio managers' favoured stocks (for instance managers who want to buy stocks highly correlated to Apple).
- ii) Find out "highly" influential stocks.
 - **Measure:** Vertices (Stocks) "strongly correlated" with most stocks. For instance, this could be represented by stocks with the highest number of degrees of a high correlation level setting on the slider bar (e.g. 0.6-1).
 - **Practical Application:** These are stocks with high correlations to the most number of peers. For risk hedging: A large number of highly influential stocks with outsized movements in a current market session could mean volatile sessions in the upcoming later markets.
- iii) Find out "most stable" stocks with least correlation to other stocks.
 - **Measure:** Vertices (Stocks) with "least correlated" with most stocks.
 - **Practical Application:** These will be stocks that are least affected by movements in other stocks – likely implying these could be stocks with highly specific idiosyncratic risks which are relatively immune to general market risks.
- iv) Determine stocks which are more representative of the broader market (similar to index stocks).
 - **Measure:** Vertices (Stocks) with the highest Centrality (Closeness, betweenness centrality).
 - **Practical Application:** Portfolio managers who wish to get exposure to the broad market could also choose to buy a few stocks with high centrality instead of a basket of index stocks.
- v) Determine markets/timeframes, where stocks exhibited the highest correlation.
 - **Measure:** Comparing densities across different markets or time periods.

- **Practical Application:** For example, for comparing correlations between crisis periods (Global Financial Crisis 2008-2009) vs non-crisis periods. With more metadata such as sectors/industries of stocks, (from other data extraction APIs)
 - we could also find out highly correlated regions and sectors.

H. CONCLUSION

Understanding correlations between stocks could be a useful tool for professional fund managers and investment firms holding cross-border portfolios. Such analysis could be used for short to mid-term trading opportunities, portfolio positioning or risk management purposes. Graph analytics platforms have shown to be an excellent medium for such analytics to provide an interactive query and visualisation platform.

In our case, we have utilised historical trading data from web APIs to monitor for trading patterns. Today, with the advent of high-frequency trading, vast volumes of trading data could be generated each day, depending on the frequency of data selected (real-time, daily, weekly, etc.). For a scalable solution in order to process such volumes of data in real-life usage, we adopt a big data architectural framework for our overall workflow.

We have highlighted several use cases above for such analysis in our presentation. Nonetheless, applications of such analysis are wide-ranging and ingestion, processing frequency and storage methods could be varied depending on the use case.

I. APPENDIX

Codes below also included as separate files attached along with this document

1. PySpark codes: Correlation computation and conversion to graph data

```
import pandas as pd
import numpy as np

# Load the dataset
data =
spark.read.csv("wasb://datastore1@bigdataclustehdistorage1.blob.core.windows.net/stockdata_
small.csv", header = True, mode = "DROPMALFORMED", inferSchema = True)

# Convert into pandas Dataframe
df = data.toPandas()

# Create Vertices Dataframe
date = df['Date'].max()
df2 = df[df['Date'] == date]
node = df2.loc[:,['Ticker', 'Name', 'Price']]
node.columns = ['id', 'id_name', 'id_price']
node.head(10)

# Convert into wide format table then again from table to pandas dataframe
dfwide = df.pivot(index='Date', columns = 'Ticker', values = 'PxChange')
df = pd.DataFrame(dfwide.to_records())

# Drop the first Date column for Correlation
df_drop = df.drop(['Date'], axis=1)

# Find the correlation between stocks
df_corr = df_drop.corr()

# Convert the correlation dataframe into matrix
matrix = df_corr.as_matrix()

# Find the number of rows and columns of the matrix
count_row = df_corr.shape[0]
count_col = df_corr.shape[1]

# Create empty Adjacency Data Frame with 3 columns as TickerA, TickerB and Correlation
adj_df = pd.DataFrame(columns=['TickerAA', 'dst', 'weight'])

# Create a node dataframe for graph frame
ticker = list(df_corr.index)

# Store the correlation values in the Adjacency Data Frame from correlation matrix
i = 0
count = 0;
for i in range(count_row):
    j = 0
    for j in range(i):
        if(j < i and j < count_col):
            adj_df.loc[count, "src"] = ticker[i]
            adj_df.loc[count, "dst"] = ticker[i-j-1]
            adj_df.loc[count, "weight"] = matrix[i,j]
            count = count + 1
```

```

adj_df.shape # see Adjacency Dataframe shape

# Remove the column with NaN values
adj_df_drop = adj_df.drop(['TickerAA'], axis=1)

# Convert the correlation column to float
adj_df_drop["weight"] = adj_df_drop["weight"].astype(float)

# Two new columns to Adjacency Dataframe :- abs_weight and Color
adj_df_drop['abs_weight'] = abs(adj_df_drop['weight'])
adj_df_drop['Color'] = np.where(adj_df_drop['weight'] > 0, 'Blue', 'Red')

# Threshold value for correlation is 0.6
#adj_df_drop['abs_weight'] = np.where(adj_df_drop['abs_weight'] < 0.6,
0,adj_df_drop['abs_weight'])
#temp = adj_df_drop[adj_df_drop['abs_weight'] >= 0.6]

# Rearrange Adjacency Dataframe columns
new_order = [2,0,1,3,4]
#adjacency_df = temp[temp.columns[new_order]]
adjacency_df = adj_df_drop[adj_df_drop.columns[new_order]]

adjacency_df.head(10)

# Convert Link and Node dataframe into spark dataframe
edge = spark.createDataFrame(adjacency_df)
node = spark.createDataFrame(node)

# Save it into blob storage container
edge.repartition(1)\
.write.format("csv")\
.option("header", True)\
.mode("overwrite")\
.save("wasb://datastore1@bigdataclustehdistorage1.blob.core.windows.net/edge",header =
'true')

node.repartition(1)\
.write.format("csv")\
.option("header", True)\
.mode("overwrite")\
.save("wasb://datastore1@bigdataclustehdistorage1.blob.core.windows.net/node",header =
'true')

```

2. R code: Graph analytics

```

library(tidyverse)
library(RColorBrewer)
# display.brewer.all()
library(igraph)

#install.packages("AzureStor")
library(AzureStor)

## Download files directly from Azure Blob Storages into working directory
# nodes.csv
download_from_url(

```

```

"https://bigdataissstorageasia.blob.core.windows.net/graphnodes/nodes.csv",
"nodes.csv",

key="bbtD6brqITz4QHFU2tR4VZG5dN3DvAewjyuyHt/6+BpgUJrKcVCC4iJtZNtlNL0pYSJDhg4F
Lg0Avf3e+uJoUA==",
  overwrite=T)

# edges.csv
download_from_url(
  "https://bigdataissstorageasia.blob.core.windows.net/graphedges/edges.csv",
  "edges.csv",

key="bbtD6brqITz4QHFU2tR4VZG5dN3DvAewjyuyHt/6+BpgUJrKcVCC4iJtZNtlNL0pYSJDhg4F
Lg0Avf3e+uJoUA==",
  overwrite=T)

node <- read.csv("nodes.csv", header = TRUE, sep = ",")
link <- read.csv("edges.csv", header = TRUE)
link$direction = if_else(link$weight > 0, 1, -1)
link$weight = abs(link$weight)

## filter off edges with correlation < 0.6
link <- link %>%
  filter(abs_weight > 0.6)

g <- graph.data.frame(link, node, directed = F)
g

plot(g,
  layout = layout.circle, # default layout.auto, lgf)
  edge.color = E(g)$Color,
  edge.width = E(g)$abs_weight*2,
  edge.curved = 0.5,

  vertex.frame.color = "deepskyblue4",
  vertex.color=rgb(red=0.1,green=0.6,blue=0.8,alpha=0.8),
  vertex.label.family = "Arial",
  vertex.size = 1, #default 15
  vertex.label.font = 1, # 1: for plain text
  vertex.label.cex = 0.8 # Font size
)

# 1. Maximal Clique
largest_cliques(g)
#[[1]]
#+ 4/30 vertices, named, from 4cd411f:
# [1] 1398.HK 000333.SZ 000651.SZ 0386.HK

#[[2]]
#+ 4/30 vertices, named, from 4cd411f:
# [1] 000002.SZ 000651.SZ 0883.HK 2628.HK

#2&3. Vertices with highest degree and lowest degree
V(g)$name[degree(g)==max(degree(g))] # [1] "000002.SZ"
V(g)$name[degree(g)==min(degree(g))] # [1] "RDSA.L"

#4. Vertices with highest centrality
degree.cent <- centr_degree(g, mode = "all")

```



```

degree.cent$res
# [1] 3 3 3 8 8 4 5 9 7 1 5 3 10 7 9 3 6 2 4 7 3 9 5 7 2 5 7 9 2 4

# Highest and lowest closeness
V(g)$name[closeness(g, mode="all")==max(closeness(g, mode="all"))] # [1] "6098.T"
V(g)$name[closeness(g, mode="all")==min(closeness(g, mode="all"))] # [1] "RDSA.L"

# Highest and lowest betweenness
V(g)$name[betweenness(g)==max(betweenness(g))] # [1] "6098.T"
V(g)$name[betweenness(g)==min(betweenness(g))] # [1] "SIRI" "RDSA.L" "1093.HK"
"9984.T" "TSCO.L"

V(g)$name[edge_betweenness(g)==max(edge_betweenness(g))] # [1] "1398.HK"
V(g)$name[edge_betweenness(g)==min(edge_betweenness(g))] # [1] NA

#5. Density of a graph
edge_density(g, loops=TRUE) # [1] 0.17204

```