UNIVERSITY OF MALAYA

MASTER OF DATA SCIENCE
2018/2019

WQD7005
DATA MINING

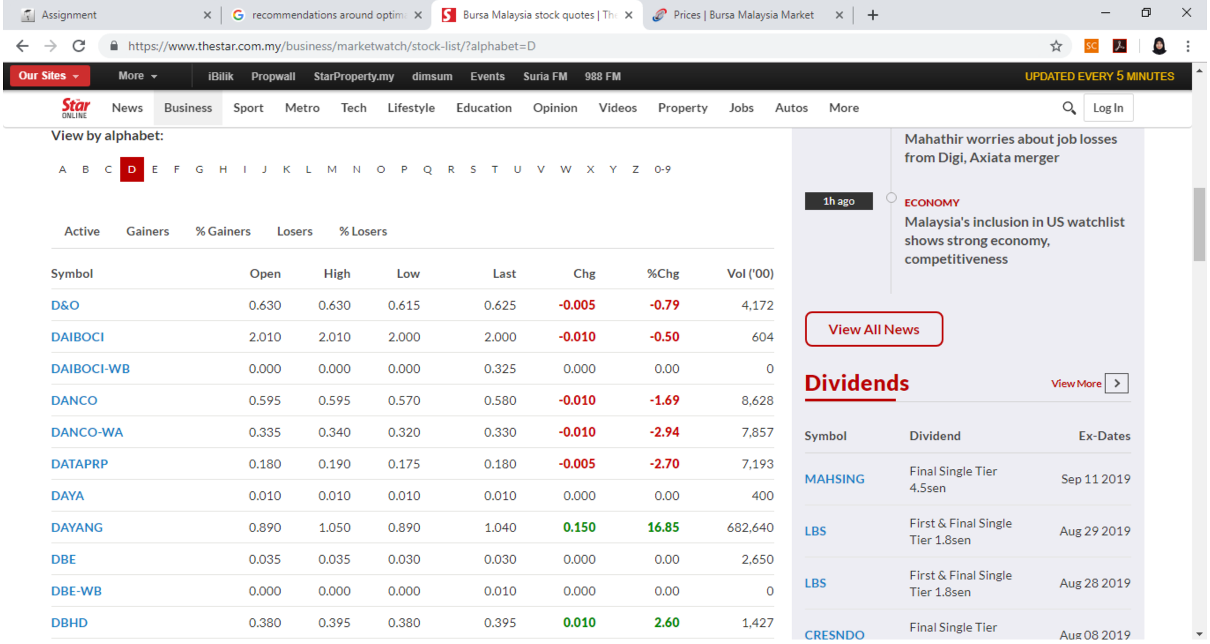
“MILESTONE 1 TO 6”

NAME: WAN NOR ANISAH BINTI WAN SHAH RAHMAN
MATRIC NUMBER: WQD170092

MILESTONE 1

Acquisition of data (Group)

In this milestone, we are required to crawl a data. The first thing to do is to identify which website need to crawl. For this project, we crawl data from "The Star.com" to get the real time stock market data. Figure 1 is the screenshot of The Star website.



Symbol	Open	High	Low	Last	Chg	%Chg	Vol ('00)
D&O	0.630	0.630	0.615	0.625	-0.005	-0.79	4,172
DAIBOCI	2.010	2.010	2.000	2.000	-0.010	-0.50	604
DAIBOCI-WB	0.000	0.000	0.000	0.325	0.000	0.00	0
DANCO	0.595	0.595	0.570	0.580	-0.010	-1.69	8,628
DANCO-WA	0.335	0.340	0.320	0.330	-0.010	-2.94	7,857
DATAPRP	0.180	0.190	0.175	0.180	-0.005	-2.70	7,193
DAYA	0.010	0.010	0.010	0.010	0.000	0.00	400
DAYANG	0.890	1.050	0.890	1.040	0.150	16.85	682,640
DBE	0.035	0.035	0.030	0.030	0.000	0.00	2,650
DBE-WB	0.000	0.000	0.000	0.010	0.000	0.00	0
DBHD	0.380	0.395	0.380	0.395	0.010	2.60	1,427

Mahathir worries about job losses from Digi, Axiata merger

1h ago

ECONOMY

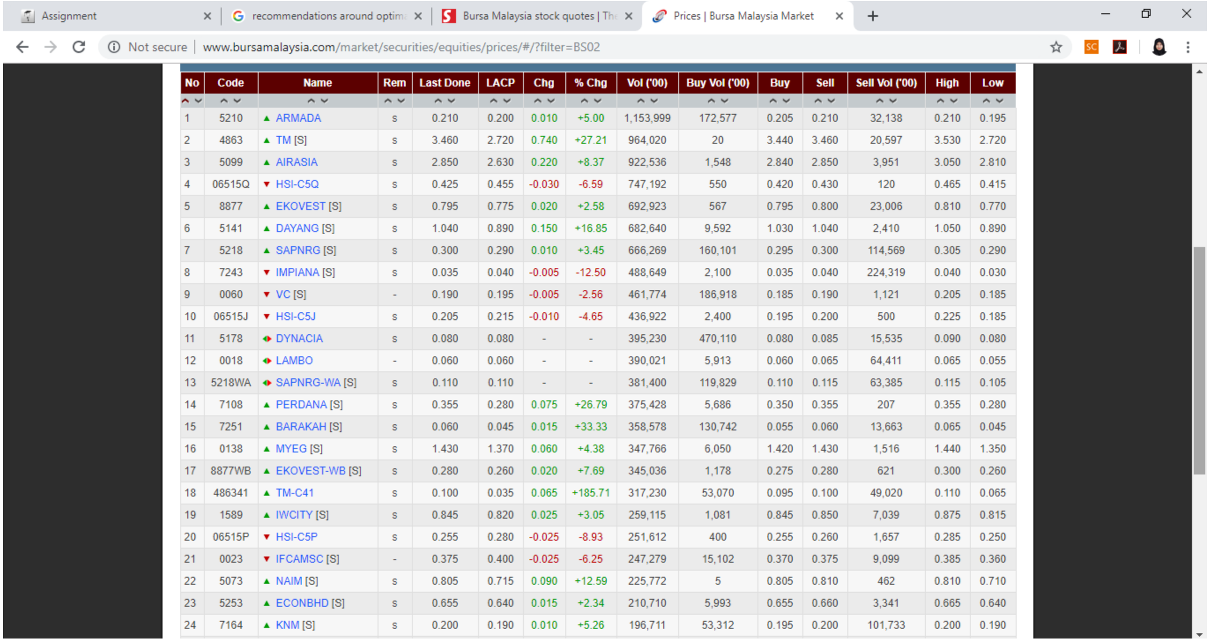
Malaysia's inclusion in US watchlist shows strong economy, competitiveness

Dividends

Symbol	Dividend	Ex-Dates
MAHSING	Final Single Tier 4.5sen	Sep 11 2019
LBS	First & Final Single Tier 1.8sen	Aug 29 2019
LBS	First & Final Single Tier 1.8sen	Aug 28 2019
CRESNDO	Final Single Tier	Aug 08 2019

Figure 1: Screenshot of The Star Website

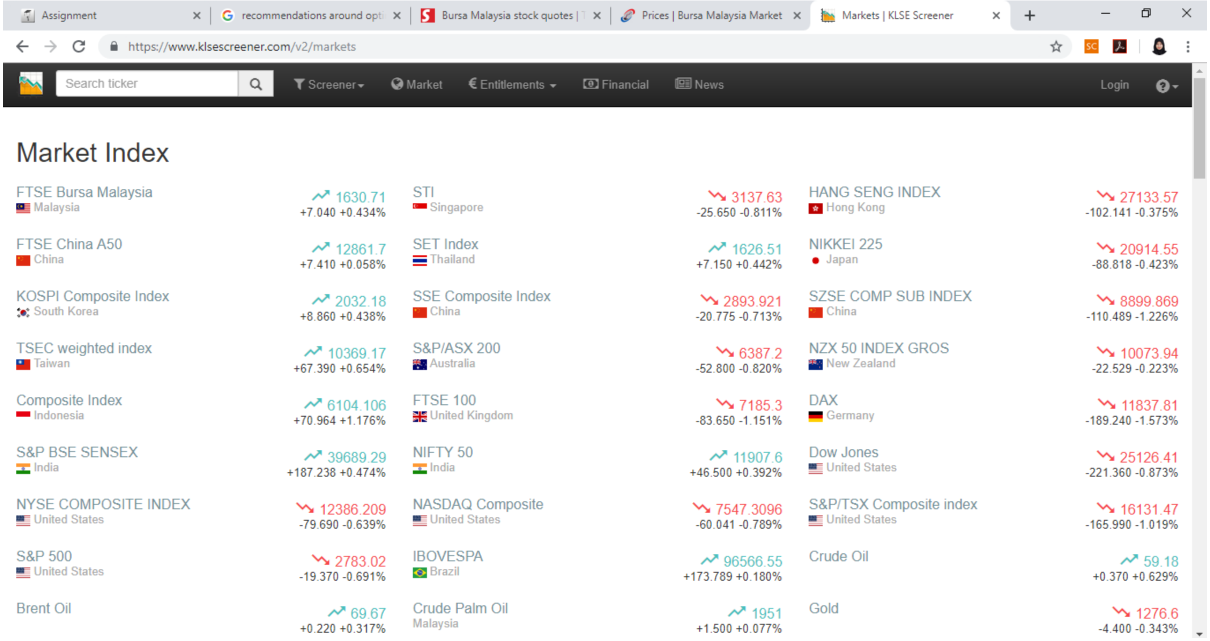
Since this website contains so many data from many companies, we try to limit the selected company based on the name on Bursa Malaysia website. Names of all company are download and saved in csv format. Then, based on these names, we crawl real time data daily for 2 weeks. Below is the screenshot for Bursa Malaysia website.


Assignment
recommendations around optim
Bursa Malaysia stock quotes | Th
Prices | Bursa Malaysia Market
Not secure
www.bursamalaysia.com/market/securities/equities/prices/#/?filter=BS02

No	Code	Name	Rem	Last Done	LACP	Chg	% Chg	Vol ('00)	Buy Vol ('00)	Buy	Sell	Sell Vol ('00)	High	Low
1	5210	▲ ARMADA	s	0.210	0.200	0.010	+5.00	1,153,999	172,577	0.205	0.210	32,138	0.210	0.195
2	4863	▲ TM [S]	s	3.460	2.720	0.740	+27.21	964,020	20	3.440	3.460	20,597	3.530	2.720
3	5099	▲ AIRASIA	s	2.850	2.630	0.220	+8.37	922,536	1,548	2.840	2.850	3,951	3.050	2.810
4	06515Q	▼ HSI-C5Q	s	0.425	0.455	-0.030	-6.59	747,192	550	0.420	0.430	120	0.465	0.415
5	8877	▲ EKOVEST [S]	s	0.795	0.775	0.020	+2.58	692,923	567	0.795	0.800	23,006	0.810	0.770
6	5141	▲ DAYANG [S]	s	1.040	0.890	0.150	+16.85	682,640	9,592	1.030	1.040	2,410	1.050	0.890
7	5218	▲ SAPNRG [S]	s	0.300	0.290	0.010	+3.45	666,269	160,101	0.295	0.300	114,569	0.305	0.290
8	7243	▼ IMPIANA [S]	s	0.035	0.040	-0.005	-12.50	488,649	2,100	0.035	0.040	224,319	0.040	0.030
9	0060	▼ VC [S]	-	0.190	0.195	-0.005	-2.56	461,774	186,918	0.185	0.190	1,121	0.205	0.185
10	06515J	▼ HSI-C5J	s	0.205	0.215	-0.010	-4.65	436,922	2,400	0.195	0.200	500	0.225	0.185
11	5178	◆ DYNACIA	s	0.080	0.080	-	-	395,230	470,110	0.080	0.085	15,535	0.090	0.080
12	0018	◆ LAMBO	-	0.060	0.060	-	-	390,021	5,913	0.060	0.065	64,411	0.065	0.055
13	5218WA	◆ SAPNRG-WA [S]	s	0.110	0.110	-	-	381,400	119,829	0.110	0.115	63,385	0.115	0.105
14	7108	▲ PERDANA [S]	s	0.355	0.280	0.075	+26.79	375,428	5,686	0.350	0.355	207	0.355	0.280
15	7251	▲ BARAKAH [S]	s	0.060	0.045	0.015	+33.33	358,578	130,742	0.055	0.060	13,663	0.065	0.045
16	0138	▲ MYEG [S]	s	1.430	1.370	0.060	+4.38	347,766	6,050	1.420	1.430	1,516	1.440	1.350
17	8877WB	▲ EKOVEST-WB [S]	s	0.280	0.260	0.020	+7.69	345,036	1,178	0.275	0.280	621	0.300	0.260
18	486341	▲ TM-C41	s	0.100	0.035	0.065	+185.71	317,230	53,070	0.095	0.100	49,020	0.110	0.065
19	1589	▲ IWCITY [S]	s	0.845	0.820	0.025	+3.05	259,115	1,081	0.845	0.850	7,039	0.875	0.815
20	06515P	▼ HSI-C5P	s	0.255	0.280	-0.025	-8.93	251,612	400	0.255	0.260	1,657	0.285	0.250
21	0023	▼ IFCAMSC [S]	-	0.375	0.400	-0.025	-6.25	247,279	15,102	0.370	0.375	9,099	0.385	0.360
22	5073	▲ NAIM [S]	s	0.805	0.715	0.090	+12.59	225,772	5	0.805	0.810	462	0.810	0.710
23	5253	▲ ECONBHD [S]	s	0.655	0.640	0.015	+2.34	210,710	5,993	0.655	0.660	3,341	0.665	0.640
24	7164	▲ KNM [S]	s	0.200	0.190	0.010	+5.26	196,711	53,312	0.195	0.200	101,733	0.200	0.190

Figure 2: Screenshot of Bursa Malaysia Website

The third website we used to obtain our data is from KLSEscreener.com. This is the website that we used to get stock market data for Annual and Quarter. Annual is a dataset contains stock market data based on year while Quarter is based on month.

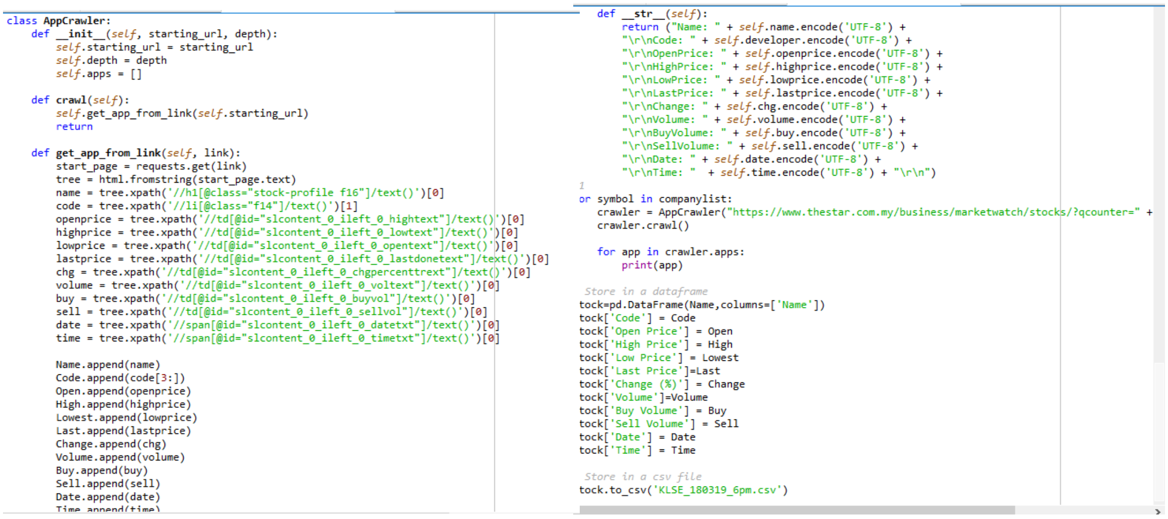

Assignment
recommendations around opt
Bursa Malaysia stock quotes |
Prices | Bursa Malaysia Market
Markets | KLSE Screener
https://www.klsescreener.com/v2/markets
Search ticker
Screener
Market
Entitlements
Financial
News
Login

Market Index

FTSE Bursa Malaysia Malaysia	1630.71 +7.040 +0.434%	STI Singapore	3137.63 -25.650 -0.811%	HANG SENG INDEX Hong Kong	27133.57 -102.141 -0.375%
FTSE China A50 China	12861.7 +7.410 +0.058%	SET Index Thailand	1626.51 +7.150 +0.442%	NIKKEI 225 Japan	20914.55 -88.818 -0.423%
KOSPI Composite Index South Korea	2032.18 +8.860 +0.438%	SSE Composite Index China	2893.921 -20.775 -0.713%	SZSE COMP SUB INDEX China	8899.869 -110.489 -1.226%
TSEC weighted index Taiwan	10369.17 +67.390 +0.654%	S&P/ASX 200 Australia	6387.2 -52.800 -0.820%	NZX 50 INDEX GROS New Zealand	10073.94 -22.529 -0.223%
Composite Index Indonesia	6104.106 +70.964 +1.176%	FTSE 100 United Kingdom	7185.3 -83.650 -1.151%	DAX Germany	11837.81 -189.240 -1.573%
S&P BSE SENSEX India	39689.29 +187.238 +0.474%	NIFTY 50 India	11907.6 +46.500 +0.392%	Dow Jones United States	25126.41 -221.360 -0.873%
NYSE COMPOSITE INDEX United States	12386.209 -79.690 -0.639%	NASDAQ Composite United States	7547.3096 -60.041 -0.789%	S&P/TSX Composite index United States	16131.47 -165.990 -1.019%
S&P 500 United States	2783.02 -19.370 -0.691%	IBOVESPA Brazil	96566.55 +173.789 +0.180%	Crude Oil	59.18 +0.370 +0.629%
Brent Oil	69.67 +0.220 +0.317%	Crude Palm Oil Malaysia	1951 +1.500 +0.077%	Gold	1276.6 -4.400 -0.343%

Figure 3: Screenshot of KLSEscreener.

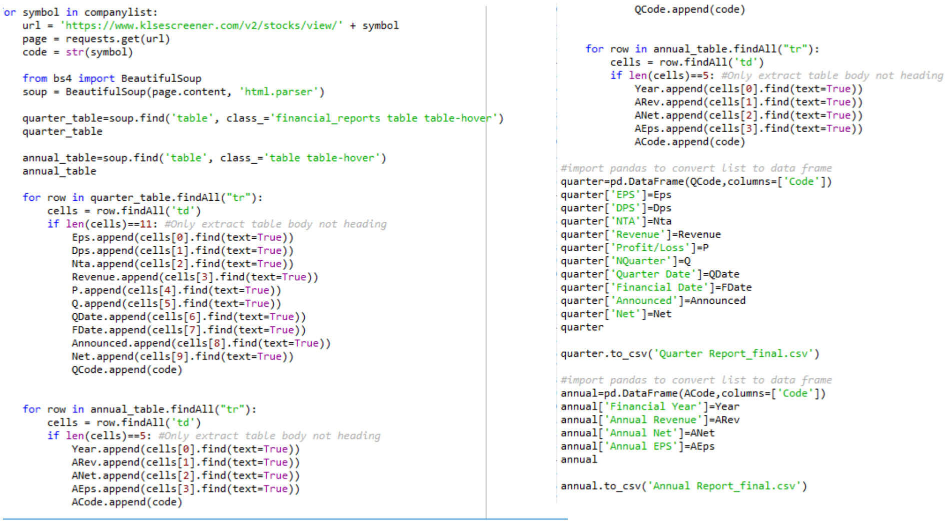
In order to crawl all the data, we use python code run with spyder. Three different code need to be created since we are going to crawl three different datasets. For annual and quarter dataset, we just need to crawl them once while for daily dataset, we crawl it every day for two weeks. Below are the screenshots for all the codes.



```
class AppCrawler:
    def __init__(self, starting_url, depth):
        self.starting_url = starting_url
        self.depth = depth
        self.apps = []

    def crawl(self):
        self.get_app_from_link(self.starting_url)
        return

    def get_app_from_link(self, link):
        start_page = requests.get(link)
        tree = html.fromstring(start_page.text)
        name = tree.xpath('//h1[@class="stock-profile f16"]/text()')[0]
        code = tree.xpath('//li[@class="f14"]/text()')[1]
        openprice = tree.xpath('//td[@id="slcontent_0_ileft_0_hightext"]/text()')[0]
        highprice = tree.xpath('//td[@id="slcontent_0_ileft_0_lowtext"]/text()')[0]
        lowprice = tree.xpath('//td[@id="slcontent_0_ileft_0_opentext"]/text()')[0]
        lastprice = tree.xpath('//td[@id="slcontent_0_ileft_0_lastdonetext"]/text()')[0]
        chg = tree.xpath('//td[@id="slcontent_0_ileft_0_chgpercenttrext"]/text()')[0]
        volume = tree.xpath('//td[@id="slcontent_0_ileft_0_voltext"]/text()')[0]
        buy = tree.xpath('//td[@id="slcontent_0_ileft_0_buyvol"]/text()')[0]
        sell = tree.xpath('//td[@id="slcontent_0_ileft_0_sellvol"]/text()')[0]
        date = tree.xpath('//span[@id="slcontent_0_ileft_0_datetxt"]/text()')[0]
        time = tree.xpath('//span[@id="slcontent_0_ileft_0_timetxt"]/text()')[0]

        Name.append(name)
        Code.append(code[3:])
        Open.append(openprice)
        High.append(highprice)
        Lowest.append(lowprice)
        Last.append(lastprice)
        Change.append(chg)
        Volume.append(volume)
        Buy.append(buy)
        Sell.append(sell)
        Date.append(date)
        Time.append(time)
```

```
   def __str__(self):
       return ("Name: " + self.name.encode('UTF-8') +
       "\r\nCode: " + self.developer.encode('UTF-8') +
       "\r\nOpenPrice: " + self.openprice.encode('UTF-8') +
       "\r\nHighPrice: " + self.highprice.encode('UTF-8') +
       "\r\nLowPrice: " + self.lowprice.encode('UTF-8') +
       "\r\nLastPrice: " + self.lastprice.encode('UTF-8') +
       "\r\nChange: " + self.chg.encode('UTF-8') +
       "\r\nVolume: " + self.volume.encode('UTF-8') +
       "\r\nBuyVolume: " + self.buy.encode('UTF-8') +
       "\r\nSellVolume: " + self.sell.encode('UTF-8') +
       "\r\nDate: " + self.date.encode('UTF-8') +
       "\r\nTime: "  + self.time.encode('UTF-8') + "\r\n")
1
or symbol in companylist:
   crawler = AppCrawler("https://www.thestar.com.my/business/marketwatch/stocks/?qcounter=" +
   crawler.crawl()

   for app in crawler.apps:
       print(app)

 Store in a dataframe
tock=pd.DataFrame(Name,columns=['Name'])
tock['Code'] = Code
tock['Open Price'] = Open
tock['High Price'] = High
tock['Low Price'] = Lowest
tock['Last Price']=Last
tock['Change (%)'] = Change
tock['Volume']=Volume
tock['Buy Volume'] = Buy
tock['Sell Volume'] = Sell
tock['Date'] = Date
tock['Time'] = Time

 Store in a csv file
tock.to_csv('KLSE_180319_6pm.csv')
```

Figure 4: Part of python code for daily dataset.



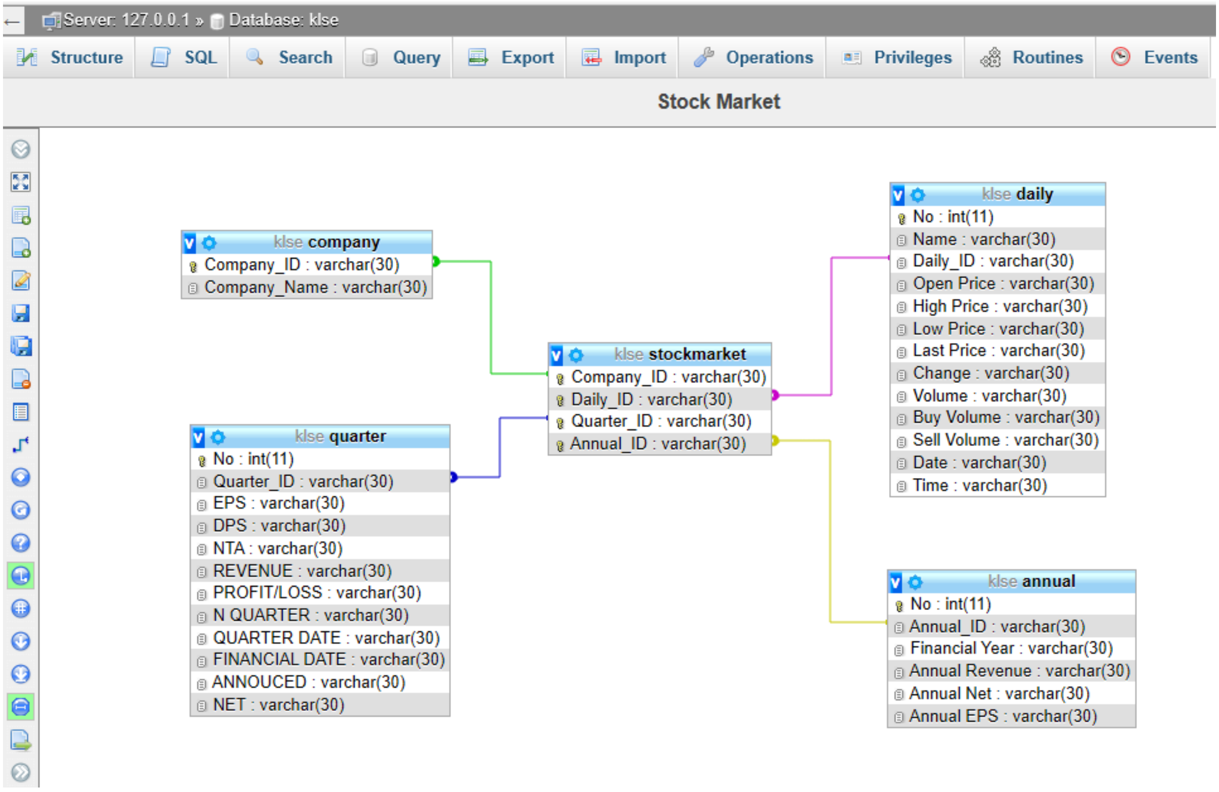
```
or symbol in companylist:
    url = 'https://www.klsescreener.com/v2/stocks/view/' + symbol
    page = requests.get(url)
    code = str(symbol)

    from bs4 import BeautifulSoup
    soup = BeautifulSoup(page.content, 'html.parser')

    quarter_table=soup.find('table', class_='financial_reports table table-hover')
    quarter_table

    annual_table=soup.find('table', class_='table table-hover')
    annual_table

    for row in quarter_table.findAll("tr"):
        cells = row.findAll('td')
        if len(cells)==11: #Only extract table body not heading
            Eps.append(cells[0].find(text=True))
            Dps.append(cells[1].find(text=True))
            Nta.append(cells[2].find(text=True))
            Revenue.append(cells[3].find(text=True))
            P.append(cells[4].find(text=True))
            Q.append(cells[5].find(text=True))
            QDate.append(cells[6].find(text=True))
            FDate.append(cells[7].find(text=True))
            Announced.append(cells[8].find(text=True))
            Net.append(cells[9].find(text=True))
            QCode.append(code)


    for row in annual_table.findAll("tr"):
        cells = row.findAll('td')
        if len(cells)==5: #Only extract table body not heading
            Year.append(cells[0].find(text=True))
            ARev.append(cells[1].find(text=True))
            ANet.append(cells[2].find(text=True))
            AEps.append(cells[3].find(text=True))
            ACode.append(code)
```

```
            QCode.append(code)


    for row in annual_table.findAll("tr"):
        cells = row.findAll('td')
        if len(cells)==5: #Only extract table body not heading
            Year.append(cells[0].find(text=True))
            ARev.append(cells[1].find(text=True))
            ANet.append(cells[2].find(text=True))
            AEps.append(cells[3].find(text=True))
            ACode.append(code)

#import pandas to convert list to data frame
quarter=pd.DataFrame(QCode,columns=['Code'])
quarter['EPS']=Eps
quarter['DPS']=Dps
quarter['NTA']=Nta
quarter['Revenue']=Revenue
quarter['Profit/Loss']=P
quarter['NQuarter']=Q
quarter['Quarter Date']=QDate
quarter['Financial Date']=FDate
quarter['Announced']=Announced
quarter['Net']=Net
quarter

quarter.to_csv('Quarter Report_final.csv')

#import pandas to convert list to data frame
annual=pd.DataFrame(ACode,columns=['Code'])
annual['Financial Year']=Year
annual['Annual Revenue']=ARev
annual['Annual Net']=ANet
annual['Annual EPS']=AEps
annual

annual.to_csv('Annual Report_final.csv')
```

Figure 5: Python code for Annual and Quarter Dataset

MILESTONE 2.
Management of data (Group).

In this milestone, we are going to do star schema in phpMyAdmin and import data into hive.



klse company
- Company_ID : varchar(30)
- Company_Name : varchar(30)

klse stockmarket
- Company_ID : varchar(30)
- Daily_ID : varchar(30)
- Quarter_ID : varchar(30)
- Annual_ID : varchar(30)

klse daily
- No : int(11)
- Name : varchar(30)
- Daily_ID : varchar(30)
- Open Price : varchar(30)
- High Price : varchar(30)
- Low Price : varchar(30)
- Last Price : varchar(30)
- Change : varchar(30)
- Volume : varchar(30)
- Buy Volume : varchar(30)
- Sell Volume : varchar(30)
- Date : varchar(30)
- Time : varchar(30)

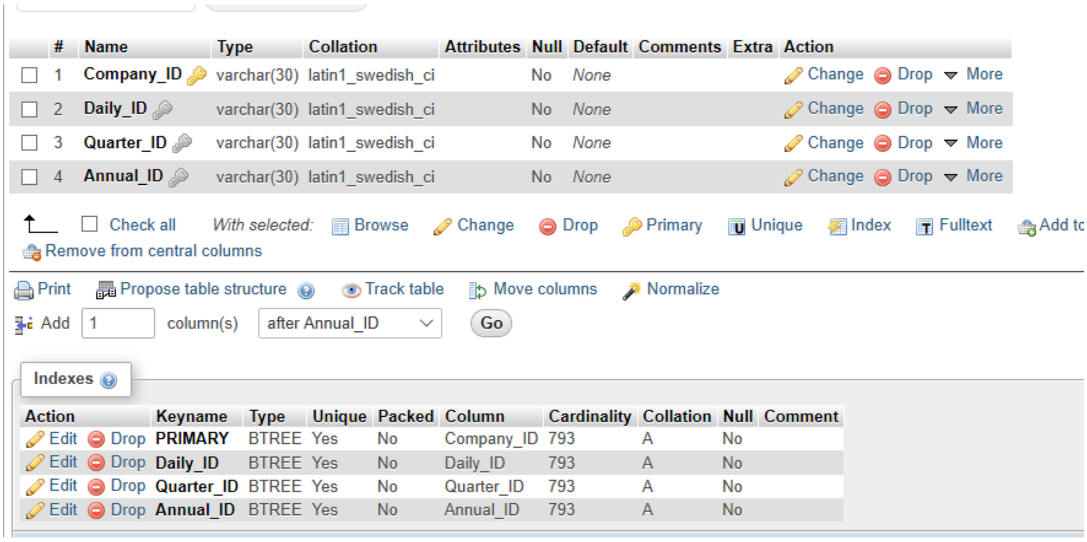
klse quarter
- No : int(11)
- Quarter_ID : varchar(30)
- EPS : varchar(30)
- DPS : varchar(30)
- NTA : varchar(30)
- REVENUE : varchar(30)
- PROFIT/LOSS : varchar(30)
- N QUARTER : varchar(30)
- QUARTER DATE : varchar(30)
- FINANCIAL DATE : varchar(30)
- ANNOUCED : varchar(30)
- NET : varchar(30)

klse annual
- No : int(11)
- Annual_ID : varchar(30)
- Financial Year : varchar(30)
- Annual Revenue : varchar(30)
- Annual Net : varchar(30)
- Annual EPS : varchar(30)

Figure 1: Star schema screenshot for stock market data.

Figure 1 is the star schema for stock market datasets which is being saved in phpMyAdmin. When all the datasets are uploaded, the design of star schema is created in designer page. For star schema, there are two types of table which are fact table and the dimensional tables. Fact table is the table located in the middle that connected with the tables around it, called dimension tables. In this case, the fact table's name is *stockmarket* and the dimensions table's name are *company, daily, quarter, and annual.*

For fact table, there are four attributes which are *company_ID, Daily_ID, Quarter_ID and Annual_ID.* In order to connect those attributes to another attributes in dimension table, they need to be assigned to a key. In this case, all the attributes contain special character. Which means, all the character in those attributes are different from one another. Thus, primary key can be used. However, one table only can have one primary key. For that reason, *Company_ID* hold primary key for this table. Other tables who have a special character too are assigned as a unique in order to allow them connected to other tables. Unique is just like a primary key which only can contains special character but, they can be many unique in one table. Figure 2 below is the screenshot of the stock market's table structure.



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**Company_ID**	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
□	2	**Daily_ID**	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
□	3	**Quarter_ID**	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
□	4	**Annual_ID**	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More

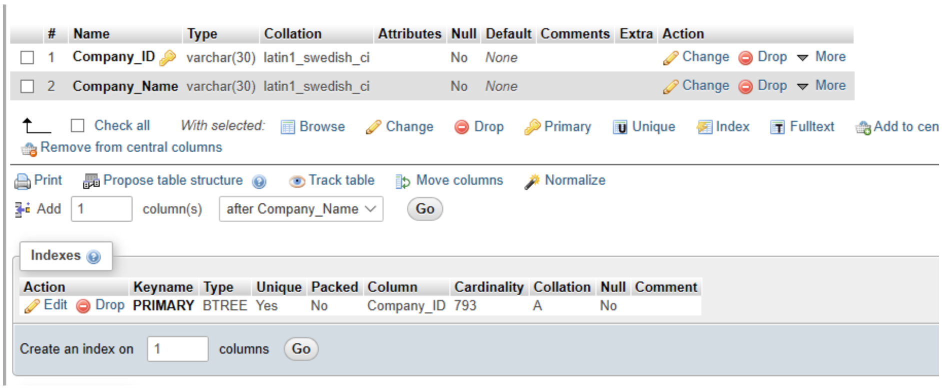
□ Check all *With selected:* Browse Change Drop Primary Unique Index Fulltext Add tc
Remove from central columns
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after Annual_ID Go
Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	**PRIMARY**	BTREE	Yes	No	Company_ID	793	A	No	
Edit Drop	**Daily_ID**	BTREE	Yes	No	Daily_ID	793	A	No	
Edit Drop	**Quarter_ID**	BTREE	Yes	No	Quarter_ID	793	A	No	
Edit Drop	**Annual_ID**	BTREE	Yes	No	Annual_ID	793	A	No	

Figure 2: Table structure of stockmarket.

Next, for dimension tables, four dimension tables are created. Same with the fact table, only one primary key can be assigned to one table. However, primary key only can contains the special or unique character which not have any duplications. Thus, primary key are used. Below are the details for every dimension tables.

a. Company



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**Company_ID**	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
□	2	**Company_Name**	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More

□ Check all *With selected:* Browse Change Drop Primary Unique Index Fulltext Add to cen
Remove from central columns
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after Company_Name Go
Indexes

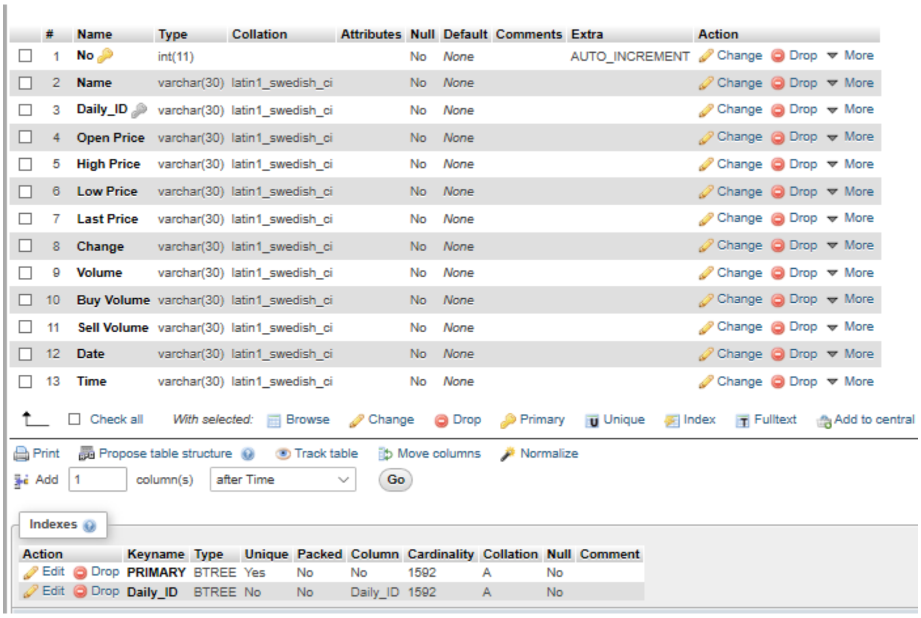
Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	**PRIMARY**	BTREE	Yes	No	Company_ID	793	A	No	

Create an index on 1 columns Go

Figure 3: Table structure of Company.

For company table, Company_ID can be assigned as a primary key since it is contains only unique character. In this attribute, all of company ID is stated. Because of this attribute is the selected attribute to be connected with fact table, foreign key is not appropriate to be assigned

b. Daily.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	No	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	Name	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
3	Daily_ID	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
4	Open Price	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
5	High Price	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
6	Low Price	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
7	Last Price	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
8	Change	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
9	Volume	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
10	Buy Volume	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
11	Sell Volume	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
12	Date	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
13	Time	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More

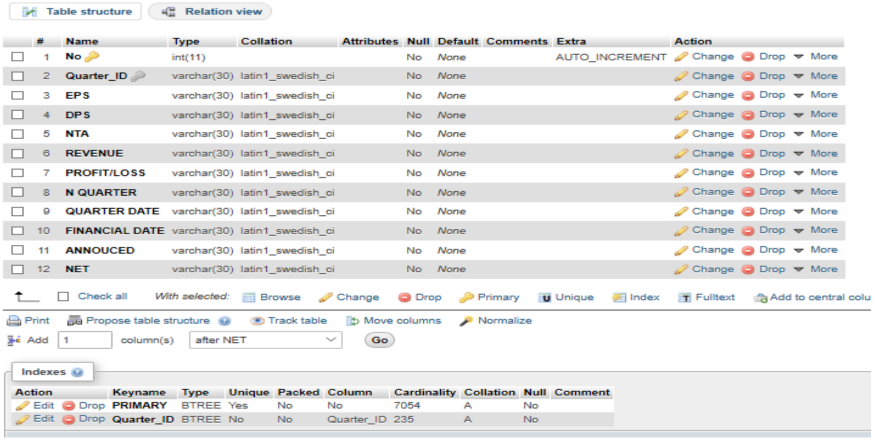
□ Check all
With selected: Browse Change Drop Primary Unique Index Fulltext Add to central
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after Time Go
Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	No	1592	A	No	
Edit Drop	Daily_ID	BTREE	No	No	Daily_ID	1592	A	No	

Figure 4: Table structure for daily.

Daily table is a bit different from company table. For this table, two types of key are assigned. Primary key is assigned to the *No* which is auto increment by the database itself. Everytime the new data import, the continuous number will be created. Thus, there are repeated number for this attribute. However, in order to connect with the fact table, foreign key is important. Daily_ID is a foreign key for this table. It contains all the daily ID of the company. But the ID is repeated for the same company.

c. Quarter.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	No	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	Quarter_ID	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
3	EPS	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
4	DPS	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
5	NTA	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
6	REVENUE	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
7	PROFIT/LOSS	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
8	N QUARTER	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
9	QUARTER DATE	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
10	FINANCIAL DATE	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
11	ANNOUCED	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
12	NET	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More

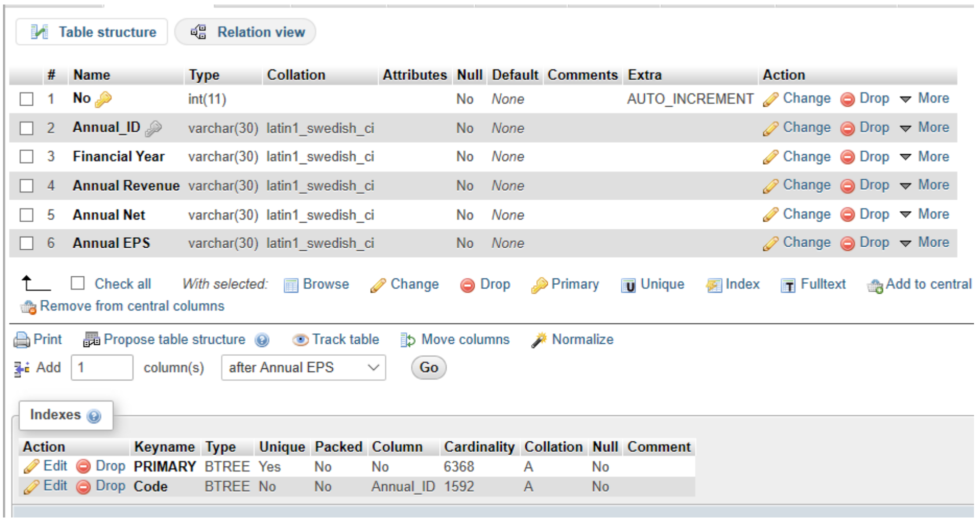
□ Check all
With selected: Browse Change Drop Primary Unique Index Fulltext Add to central colu
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after NET Go
Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	No	7054	A	No	
Edit Drop	Quarter_ID	BTREE	No	No	Quarter_ID	235	A	No	

Figure 4: Table structure for Quarter.

Quarter table is same as daily table. It contains primary key and foreign key. Primary key is assigned to the no which is auto increment to and the foreign key is assign to Quarter_ID attribute which contains the repeated ID.

d. Annual.



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	No	int(11)			No	None		AUTO_INCREMENT	Change Drop More
□	2	Annual_ID	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
□	3	Financial Year	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
□	4	Annual Revenue	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
□	5	Annual Net	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
□	6	Annual EPS	varchar(30)	latin1_swedish_ci		No	None			Change Drop More

Add 1 column(s) after Annual EPS

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	No	6368	A	No	
Edit Drop	Code	BTREE	No	No	Annual_ID	1592	A	No	

Figure 5: Table structure for Annual.

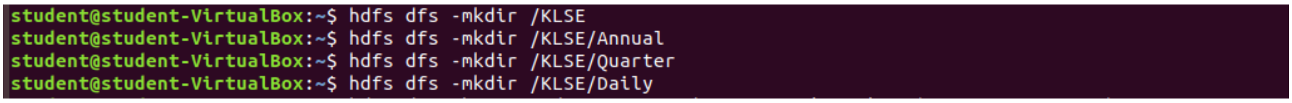
Same as quarter table, annual table contains primary key and foreign key. Primary key is assigned to the no which is auto increment to and the foreign key is assign to Annual_ID attribute which contains the repeated ID.

Roll Up (Drill-up) and Roll Down (Drill-down).

Roll up is the process of summarize data. It works by climbing up hierarchy or by dimension reduction. We can filter our data to show only the selected data. Thus, we will have a less amount of data to be shown.

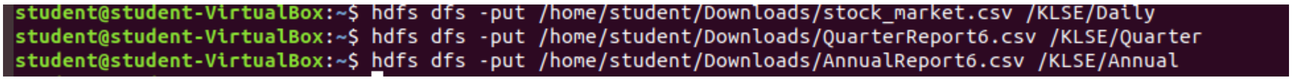
Drill down is the reverse process of Roll up. While roll up is from more detailed data to less detailed data, drill down is the process from less detailed data to more detailed data. By means, we can have more information that we need from the certain data we have by doing drill down process.

For hive, since we have three types of data, we create one main directory consist of another three sub directories names Annual, Quarter and Daily. When the directory is already created, the next step is to import all those three datasets from local host to the hdfs before they could be imported into hive table. Figure 6 and 7 is the screenshot of creating directories and importing datasets.



```
student@student-VirtualBox:~$ hdfs dfs -mkdir /KLSE
student@student-VirtualBox:~$ hdfs dfs -mkdir /KLSE/Annual
student@student-VirtualBox:~$ hdfs dfs -mkdir /KLSE/Quarter
student@student-VirtualBox:~$ hdfs dfs -mkdir /KLSE/Daily
```

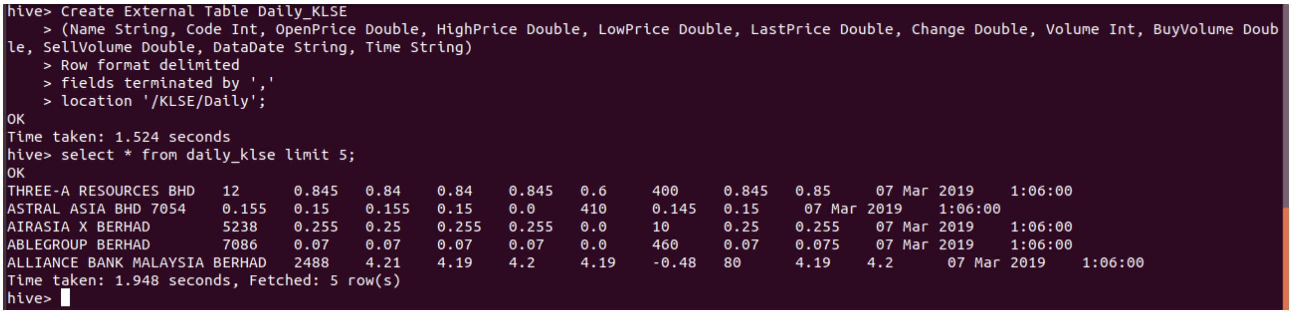
Figure 6: Create Directories.



```
student@student-VirtualBox:~$ hdfs dfs -put /home/student/Downloads/stock_market.csv /KLSE/Daily
student@student-VirtualBox:~$ hdfs dfs -put /home/student/Downloads/QuarterReport6.csv /KLSE/Quarter
student@student-VirtualBox:~$ hdfs dfs -put /home/student/Downloads/AnnualReport6.csv /KLSE/Annual
```

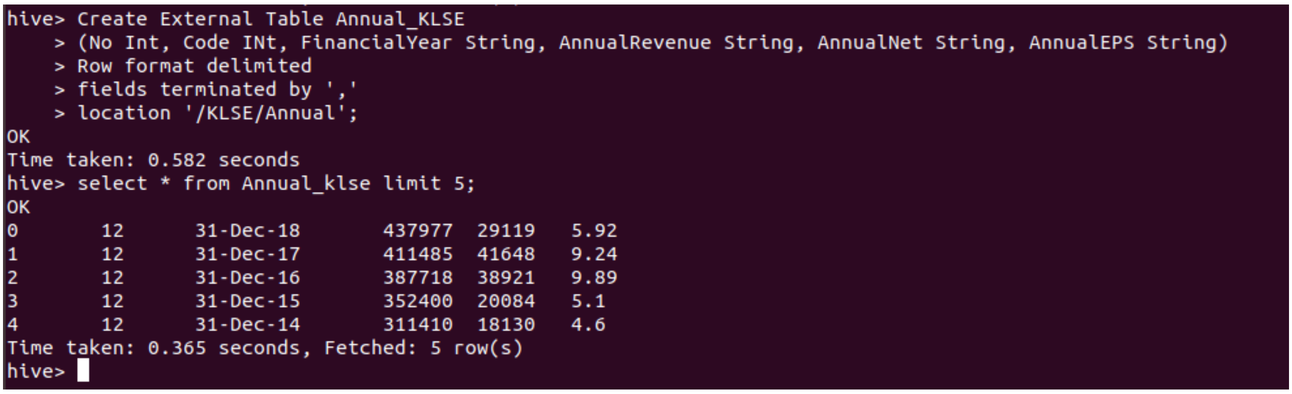
Figure 7: Import Datasets.

Then, we need to create table in hive before import datasets. Hive command is similar with mysql, thus, we can show our imported dataset directly in hive. Below are the screenshots of all the tables and their results.



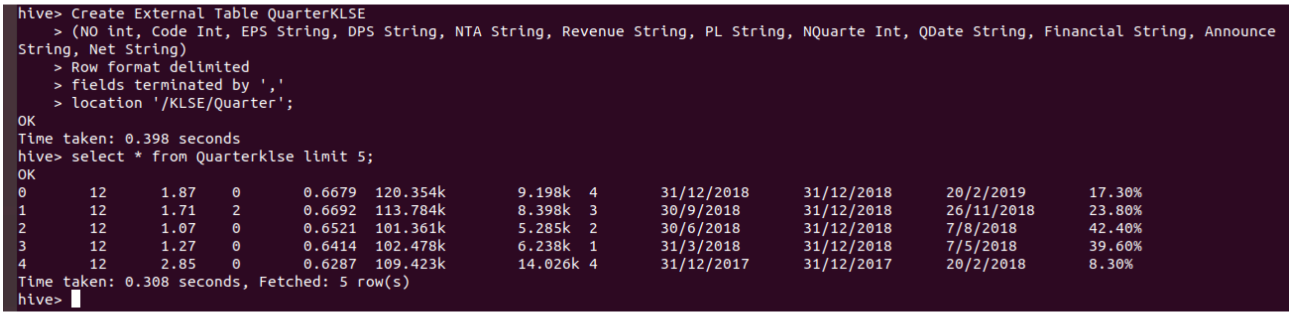
```
hive> Create External Table Daily_KLSE
    > (Name String, Code Int, OpenPrice Double, HighPrice Double, LowPrice Double, LastPrice Double, Change Double, Volume Int, BuyVolume Doub
le, SellVolume Double, DataDate String, Time String)
    > Row format delimited
    > fields terminated by ','
    > location '/KLSE/Daily';
OK
Time taken: 1.524 seconds
hive> select * from daily_klse limit 5;
OK
THREE-A RESOURCES BHD   12      0.845   0.84    0.84    0.845   0.6     400     0.845   0.85      07 Mar 2019     1:06:00
ASTRAL ASIA BHD 7054    0.155   0.15    0.155   0.15    0.0     410     0.145   0.15     07 Mar 2019     1:06:00
AIRASIA X BERHAD        5238    0.255   0.25    0.255   0.255   0.0     10      0.25    0.255     07 Mar 2019     1:06:00
ABLEGROUP BERHAD        7086    0.07    0.07    0.07    0.07    0.0     460     0.07    0.075     07 Mar 2019     1:06:00
ALLIANCE BANK MALAYSIA BERHAD   2488    4.21    4.19    4.2     4.19    -0.48   80      4.19    4.2       07 Mar 2019     1:06:00
Time taken: 1.948 seconds, Fetched: 5 row(s)
hive> 
```

Figure 8: Create and show Table for Daily Dataset



```
hive> Create External Table Annual_KLSE
    > (No Int, Code INt, FinancialYear String, AnnualRevenue String, AnnualNet String, AnnualEPS String)
    > Row format delimited
    > fields terminated by ','
    > location '/KLSE/Annual';
OK
Time taken: 0.582 seconds
hive> select * from Annual_klse limit 5;
OK
0       12      31-Dec-18       437977  29119   5.92
1       12      31-Dec-17       411485  41648   9.24
2       12      31-Dec-16       387718  38921   9.89
3       12      31-Dec-15       352400  20084   5.1
4       12      31-Dec-14       311410  18130   4.6
Time taken: 0.365 seconds, Fetched: 5 row(s)
hive> 
```

Figure 9: Create and show Table for Annual Dataset



```
hive> Create External Table QuarterKLSE
    > (NO int, Code Int, EPS String, DPS String, NTA String, Revenue String, PL String, NQuarte Int, QDate String, Financial String, Announce
String, Net String)
    > Row format delimited
    > fields terminated by ','
    > location '/KLSE/Quarter';
OK
Time taken: 0.398 seconds
hive> select * from Quarterklse limit 5;
OK
0       12      1.87    0       0.6679  120.354k        9.198k  4       31/12/2018      31/12/2018      20/2/2019       17.30%
1       12      1.71    2       0.6692  113.784k        8.398k  3       30/9/2018       31/12/2018      26/11/2018      23.80%
2       12      1.07    0       0.6521  101.361k        5.285k  2       30/6/2018       31/12/2018      7/8/2018        42.40%
3       12      1.27    0       0.6414  102.478k        6.238k  1       31/3/2018       31/12/2018      7/5/2018        39.60%
4       12      2.85    0       0.6287  109.423k        14.026k 4       31/12/2017      31/12/2017      20/2/2018       8.30%
Time taken: 0.308 seconds, Fetched: 5 row(s)
hive> 
```

Figure 10: Create and show Table for Quarter Dataset

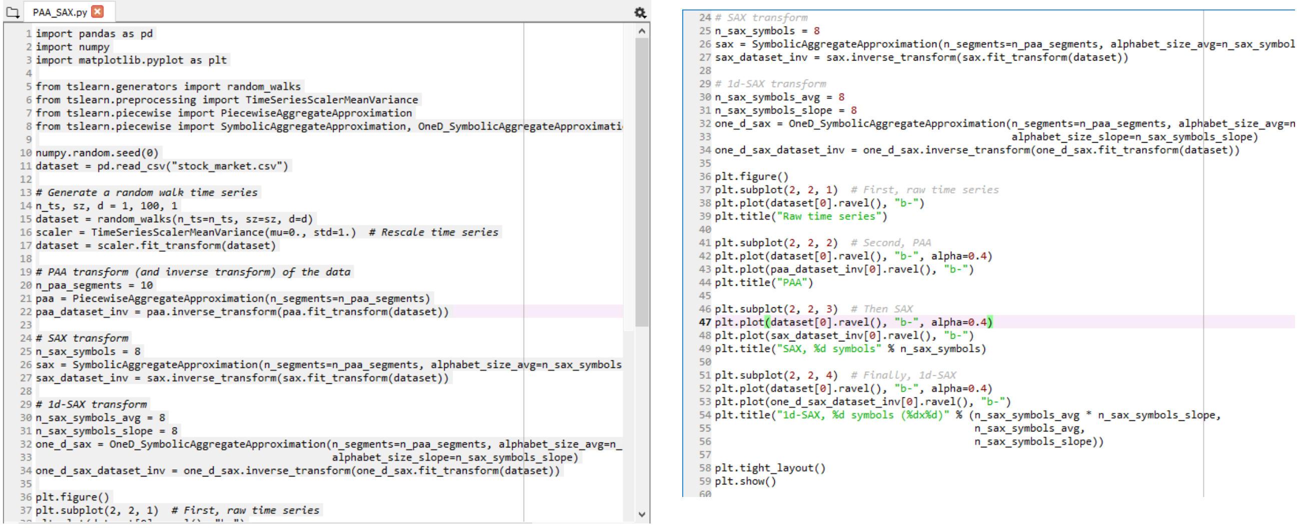
Milestone 3.
Processing of data (Group).

For this milestone, we are requested to do PAA SAX. PAA stands for Piecewise Aggregate Approximation and SAX stands for Symbolic Aggregate Approximation.

Piecewise Aggregate Approximation (PAA) is a very simple dimensionality reduction method for time series mining. It minimizes dimensionality by the mean values of equal sized frames, which misses some important information and sometimes causes inaccurate results in time series mining. While SAX is a method of discretizing time series, to better understand patterns and motifs. It involves binning the time series and then translating these discrete bins into words, which are discrete objects made of discrete letters.

We used our daily dataset as an input dataset. This dataset first needs to be processed through PAA in order to reduce the dimension of it. Then, the next step is to do SAX where the data are converted into symbol. This can help in well understanding the pattern of our data. Thus, by using this pattern, we can predict the future pattern.

Figure 1 is the code for PAA and SAX using python. Figure 2 is a comparison between PAA, SAX and 1d-SAX features.

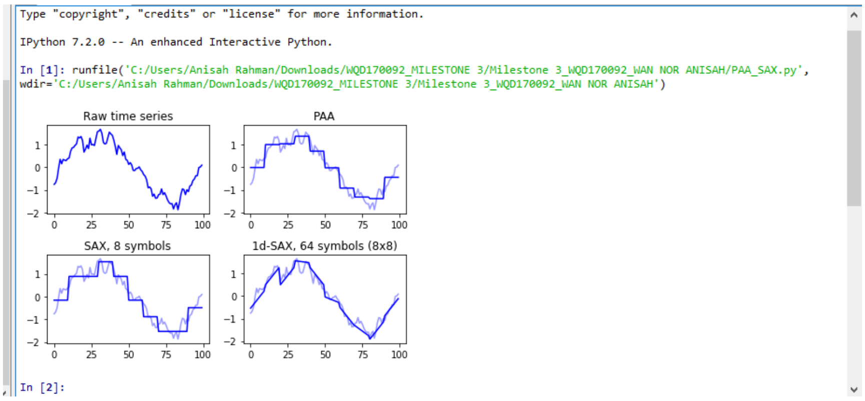


```
import pandas as pd
import numpy
import matplotlib.pyplot as plt

from tslearn.generators import random_walks
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.piecewise import PiecewiseAggregateApproximation
from tslearn.piecewise import SymbolicAggregateApproximation, OneD_SymbolicAggregateApproximati

numpy.random.seed(0)
dataset = pd.read_csv("stock_market.csv")

# Generate a random walk time series
n_ts, sz, d = 1, 100, 1
dataset = random_walks(n_ts=n_ts, sz=sz, d=d)
scaler = TimeSeriesScalerMeanVariance(mu=0., std=1.)  # Rescale time series
dataset = scaler.fit_transform(dataset)

# PAA transform (and inverse transform) of the data
n_paa_segments = 10
paa = PiecewiseAggregateApproximation(n_segments=n_paa_segments)
paa_dataset_inv = paa.inverse_transform(paa.fit_transform(dataset))

# SAX transform
n_sax_symbols = 8
sax = SymbolicAggregateApproximation(n_segments=n_paa_segments, alphabet_size_avg=n_sax_symbols
sax_dataset_inv = sax.inverse_transform(sax.fit_transform(dataset))

# 1d-SAX transform
n_sax_symbols_avg = 8
n_sax_symbols_slope = 8
one_d_sax = OneD_SymbolicAggregateApproximation(n_segments=n_paa_segments, alphabet_size_avg=n_
                                                alphabet_size_slope=n_sax_symbols_slope)
one_d_sax_dataset_inv = one_d_sax.inverse_transform(one_d_sax.fit_transform(dataset))

plt.figure()
plt.subplot(2, 2, 1)  # First, raw time series
```

```
# SAX transform
n_sax_symbols = 8
sax = SymbolicAggregateApproximation(n_segments=n_paa_segments, alphabet_size_avg=n_sax_symbol
sax_dataset_inv = sax.inverse_transform(sax.fit_transform(dataset))

# 1d-SAX transform
n_sax_symbols_avg = 8
n_sax_symbols_slope = 8
one_d_sax = OneD_SymbolicAggregateApproximation(n_segments=n_paa_segments, alphabet_size_avg=n
                                                alphabet_size_slope=n_sax_symbols_slope)
one_d_sax_dataset_inv = one_d_sax.inverse_transform(one_d_sax.fit_transform(dataset))

plt.figure()
plt.subplot(2, 2, 1)  # First, raw time series
```

```
plt.plot(dataset[0].ravel(), "b-")
plt.title("Raw time series")

plt.subplot(2, 2, 2)  # Second, PAA
plt.plot(dataset[0].ravel(), "b-", alpha=0.4)
plt.plot(paa_dataset_inv[0].ravel(), "b-")
plt.title("PAA")

plt.subplot(2, 2, 3)  # Then SAX
plt.plot(dataset[0].ravel(), "b-", alpha=0.4)
plt.plot(sax_dataset_inv[0].ravel(), "b-")
plt.title("SAX, %d symbols" % n_sax_symbols)

plt.subplot(2, 2, 4)  # Finally, 1d-SAX
plt.plot(dataset[0].ravel(), "b-", alpha=0.4)
plt.plot(one_d_sax_dataset_inv[0].ravel(), "b-")
plt.title("1d-SAX, %d symbols (%dx%d)" % (n_sax_symbols_avg * n_sax_symbols_slope,
                                          n_sax_symbols_avg,
                                          n_sax_symbols_slope))

plt.tight_layout()
plt.show()
```

Figure 1: PAA and SAX code using python



```
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Anisah Rahman/Downloads/WQD170092_MILESTONE 3/Milestone 3_WQD170092_WAN NOR ANISAH/PAA_SAX.py',
wdir='C:/Users/Anisah Rahman/Downloads/WQD170092_MILESTONE 3/Milestone 3_WQD170092_WAN NOR ANISAH')
```

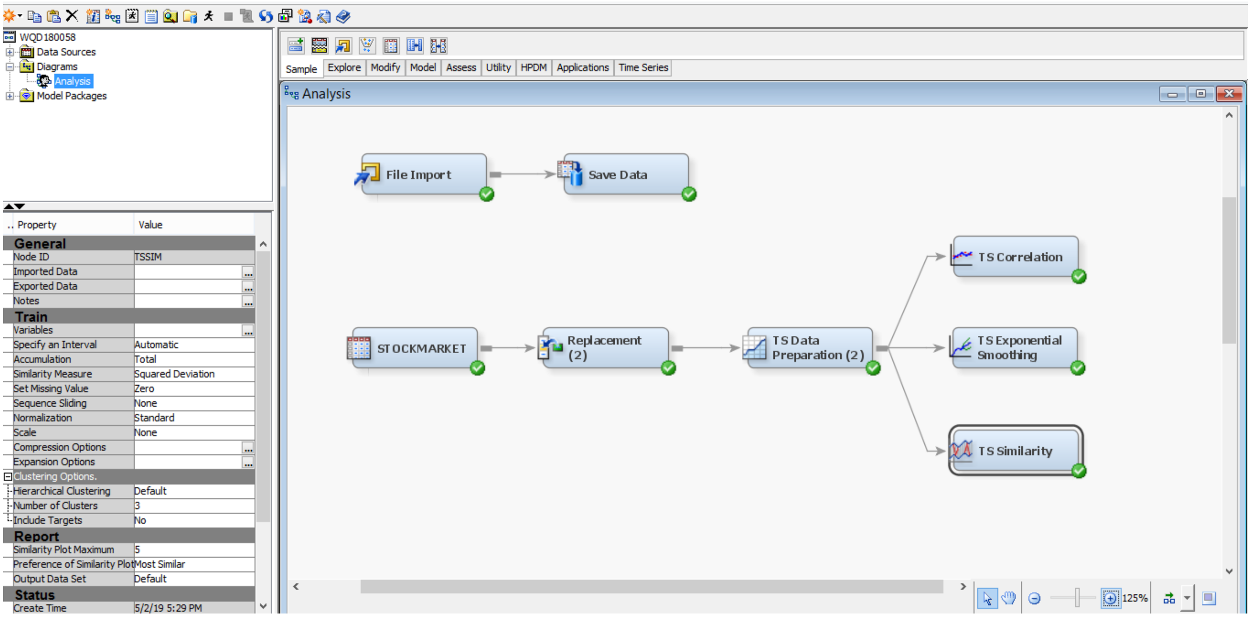
```
In [2]:
```

Figure 2: comparison between PAA, SAX and 1d-SAX features.

Milestone 4.
Interpretation of data (Individual).

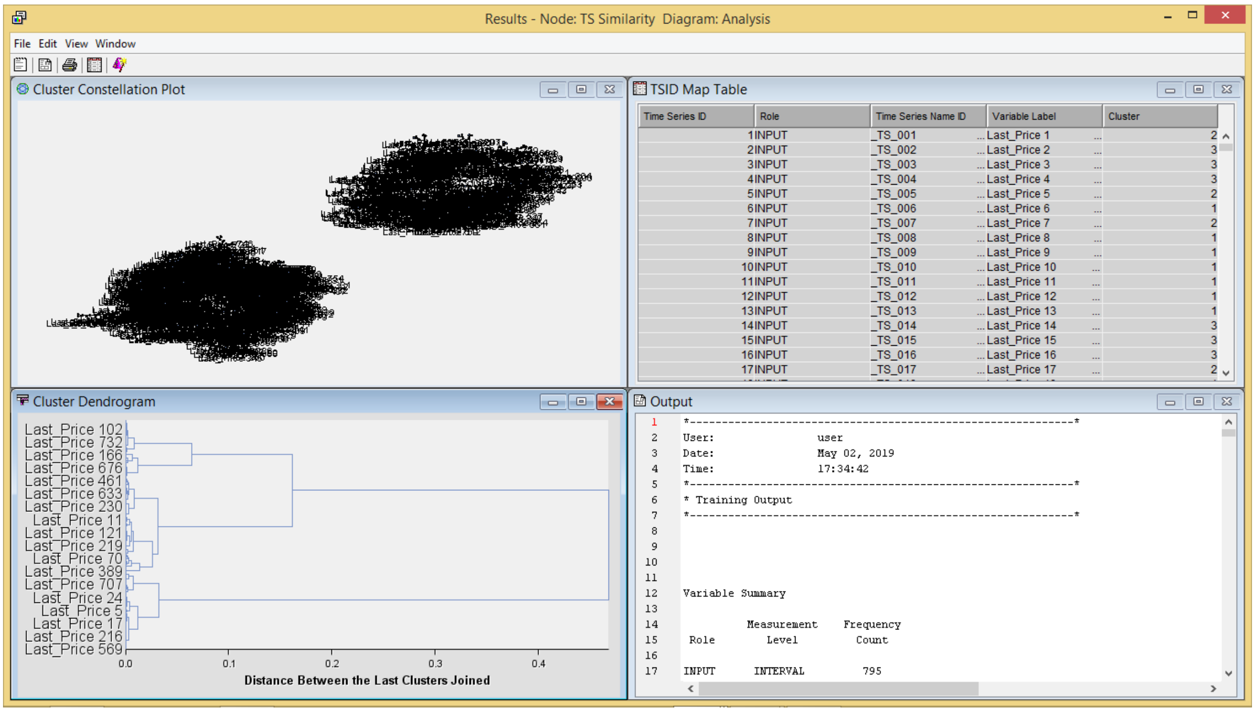
For interpretation of data, this data are clustered based on their last price similarity. From this, we can see how many clusters the data can have to ease us identify which company is the good choice to make investment.

Figure 1 is how the nodes arranged in order to get cluster for the last price data. Figure 2 is the results from TS similarity nodes. It shows the cluster plot, dendogram, map table and the output. In figure 3, shows only cluster plot.



Property	Value
General	
Node ID	TSSIM
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Specify an Interval	Automatic
Accumulation	Total
Similarity Measure	Squared Deviation
Set Missing Value	Zero
Sequence Sliding	None
Normalization	Standard
Scale	None
Compression Options	...
Expansion Options	...
Clustering Options.	
Hierarchical Clustering	Default
Number of Clusters	3
Include Targets	No
Report	
Similarity Plot Maximum	5
Preference of Similarity Plot	Most Similar
Output Data Set	Default
Status	
Create Time	5/2/19 5:29 PM

Figure 1: SAS Nodes.



TSID Map Table

Time Series ID	Role	Time Series Name ID	Variable Label	Cluster
1	INPUT	_TS_001 ...	Last_Price 1 ...	2
2	INPUT	_TS_002 ...	Last_Price 2 ...	3
3	INPUT	_TS_003 ...	Last_Price 3 ...	3
4	INPUT	_TS_004 ...	Last_Price 4 ...	3
5	INPUT	_TS_005 ...	Last_Price 5 ...	2
6	INPUT	_TS_006 ...	Last_Price 6 ...	1
7	INPUT	_TS_007 ...	Last_Price 7 ...	2
8	INPUT	_TS_008 ...	Last_Price 8 ...	1
9	INPUT	_TS_009 ...	Last_Price 9 ...	1
10	INPUT	_TS_010 ...	Last_Price 10 ...	1
11	INPUT	_TS_011 ...	Last_Price 11 ...	1
12	INPUT	_TS_012 ...	Last_Price 12 ...	1
13	INPUT	_TS_013 ...	Last_Price 13 ...	1
14	INPUT	_TS_014 ...	Last_Price 14 ...	3
15	INPUT	_TS_015 ...	Last_Price 15 ...	3
16	INPUT	_TS_016 ...	Last_Price 16 ...	3
17	INPUT	_TS_017 ...	Last_Price 17 ...	2

Output

```
*------------------------------------------------------------*
User:                user
Date:                May 02, 2019
Time:                17:34:42
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*




Variable Summary

          Measurement    Frequency
 Role        Level         Count

INPUT      INTERVAL         795
```

Figure 2: TS Similarity Diagram Analysis.

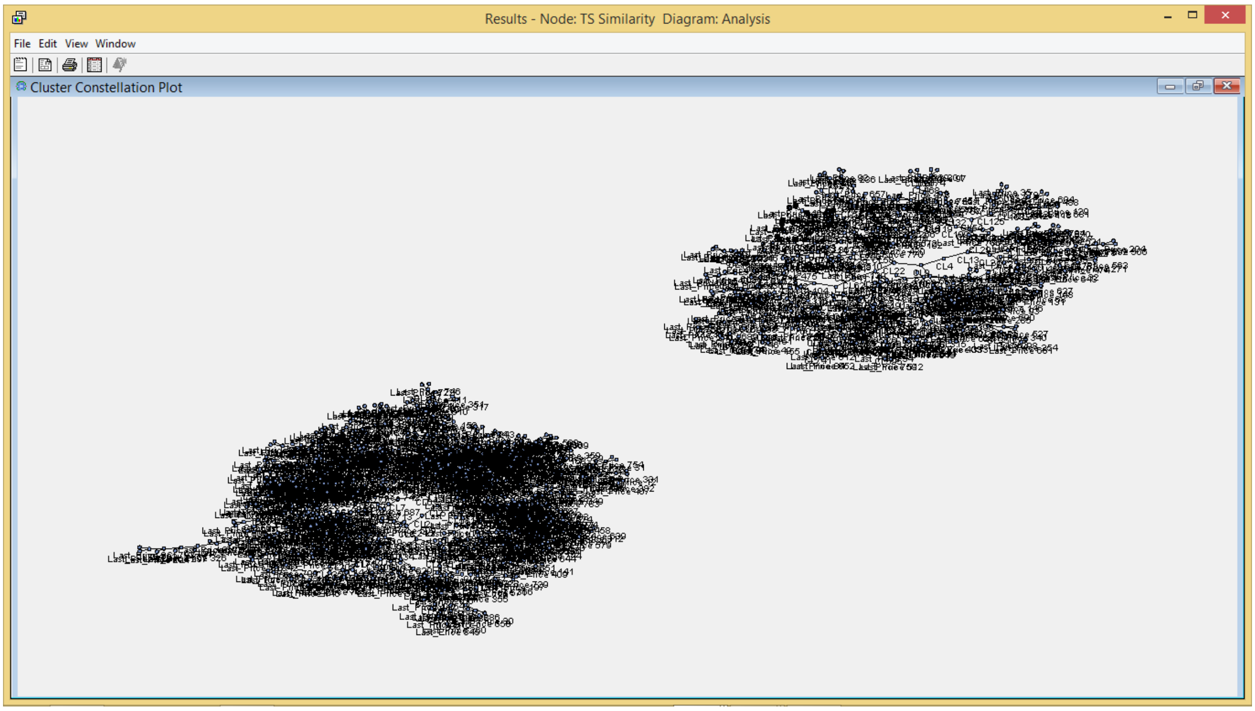
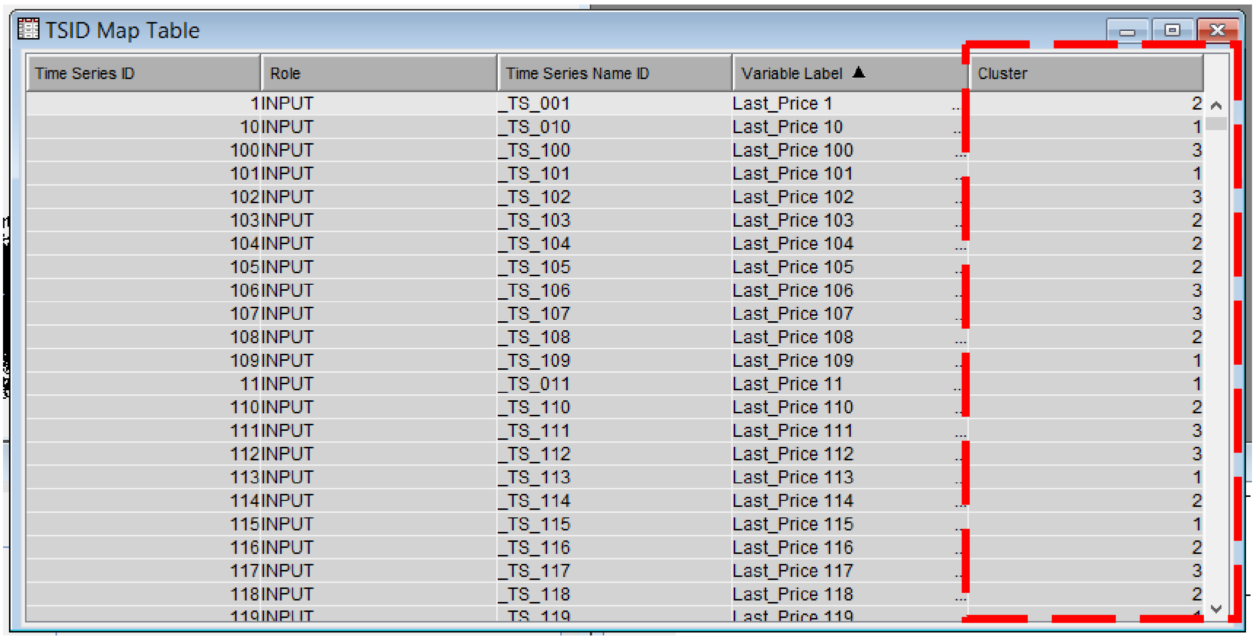

Results - Node: TS Similarity Diagram: Analysis
File Edit View Window
Cluster Constellation Plot

Figure 3: Cluster Result.

Based on the Figure 3 above, there are 2 distinct clusters that we could see. However, it is actually divided into 3 clusters as per table below.



TSID Map Table

Time Series ID	Role	Time Series Name ID	Variable Label ▲	Cluster
1	INPUT	_TS_001	Last_Price 1	2
10	INPUT	_TS_010	Last_Price 10	1
100	INPUT	_TS_100	Last_Price 100	3
101	INPUT	_TS_101	Last_Price 101	1
102	INPUT	_TS_102	Last_Price 102	3
103	INPUT	_TS_103	Last_Price 103	2
104	INPUT	_TS_104	Last_Price 104	2
105	INPUT	_TS_105	Last_Price 105	2
106	INPUT	_TS_106	Last_Price 106	3
107	INPUT	_TS_107	Last_Price 107	3
108	INPUT	_TS_108	Last_Price 108	2
109	INPUT	_TS_109	Last_Price 109	1
11	INPUT	_TS_011	Last_Price 11	1
110	INPUT	_TS_110	Last_Price 110	2
111	INPUT	_TS_111	Last_Price 111	3
112	INPUT	_TS_112	Last_Price 112	3
113	INPUT	_TS_113	Last_Price 113	1
114	INPUT	_TS_114	Last_Price 114	2
115	INPUT	_TS_115	Last_Price 115	1
116	INPUT	_TS_116	Last_Price 116	2
117	INPUT	_TS_117	Last_Price 117	3
118	INPUT	_TS_118	Last_Price 118	2
119	INPUT	_TS_119	Last_Price 119	1

Figure 4: TSID Map Table.

The cluster dendrogram below explain how the clusters are divided into three.

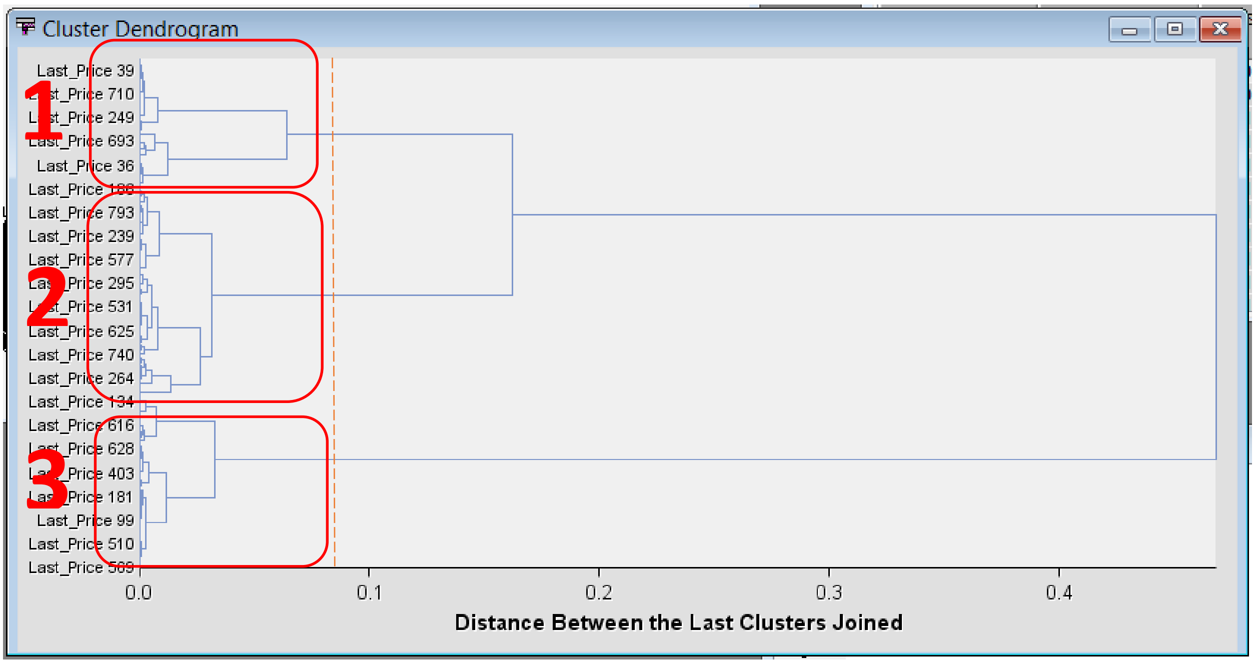
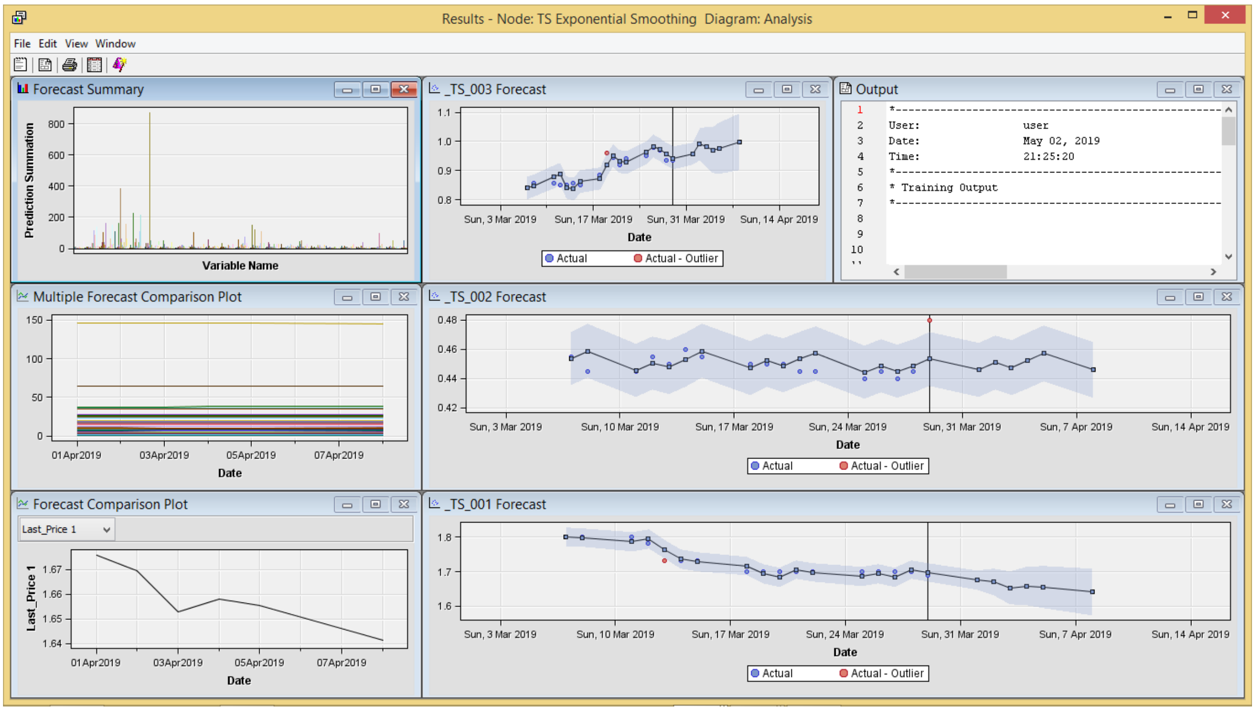


Figure 5: Cluster Dendrogram.

Milestone 5.
Communication of insights of data (Individual)

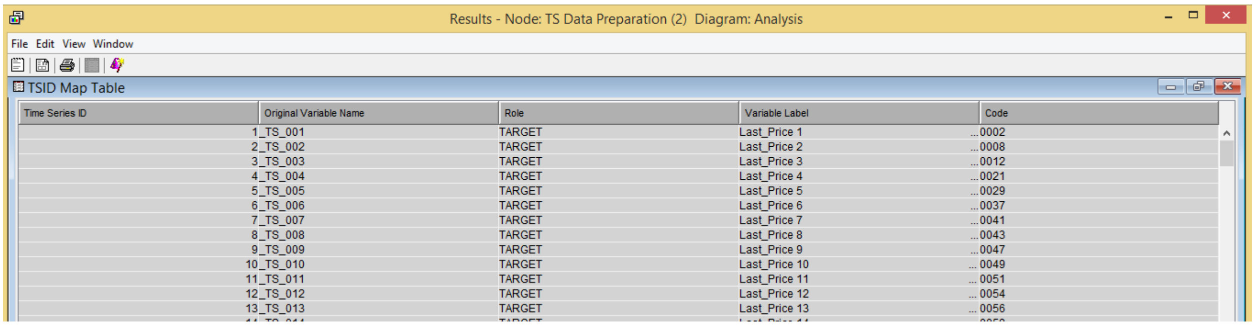
For this milestone, prediction is made. Thus, we can choose which company can be a good selection in order to make investment. Below is the result from TS Exponential Smoothing. The nodes are arranged just like what have been showed in Milestone 4 figure 1.



```
*------------------------------------------------------------
User:                  user
Date:                  May 02, 2019
Time:                  21:25:20
*------------------------------------------------------------
* Training Output
*------------------------------------------------------------
```

Figure 1: TS Exponential Smoothing Diagram Analysis.

Figure 2 below shows the description for each company (CODE) which is assign to a new TS ID.



Time Series ID	Original Variable Name	Role	Variable Label	Code
1	_TS_001	TARGET	Last_Price 1	...0002
2	_TS_002	TARGET	Last_Price 2	...0008
3	_TS_003	TARGET	Last_Price 3	...0012
4	_TS_004	TARGET	Last_Price 4	...0021
5	_TS_005	TARGET	Last_Price 5	...0029
6	_TS_006	TARGET	Last_Price 6	...0037
7	_TS_007	TARGET	Last_Price 7	...0041
8	_TS_008	TARGET	Last_Price 8	...0043
9	_TS_009	TARGET	Last_Price 9	...0047
10	_TS_010	TARGET	Last_Price 10	...0049
11	_TS_011	TARGET	Last_Price 11	...0051
12	_TS_012	TARGET	Last_Price 12	...0054
13	_TS_013	TARGET	Last_Price 13	...0056

Figure 2: TSID Map Table

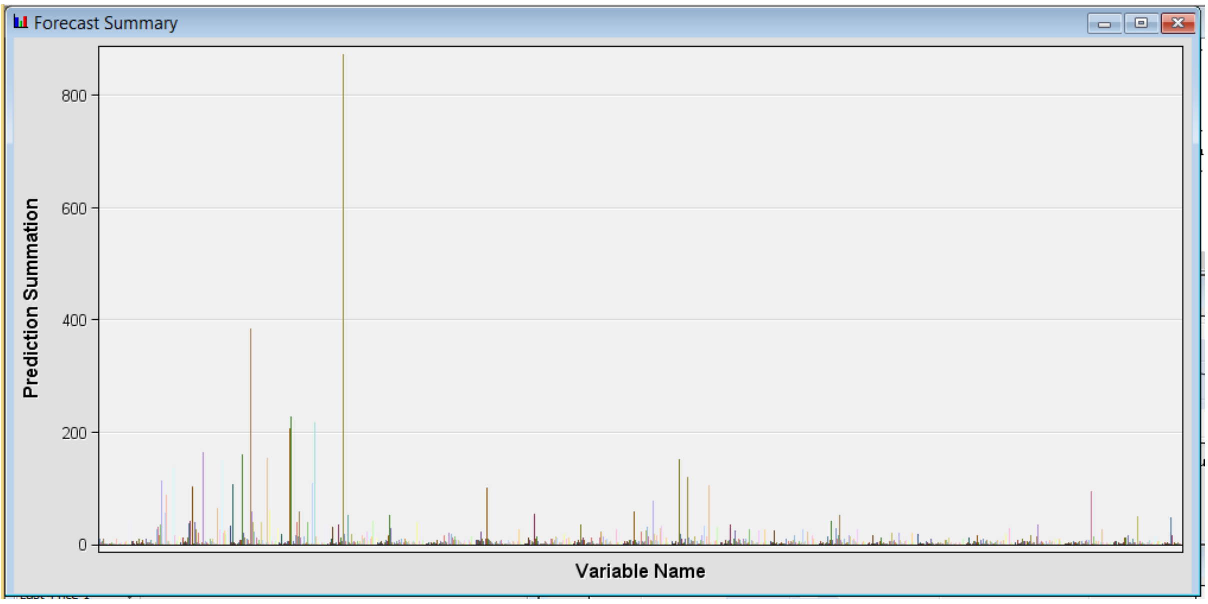


Figure 3: Forecast Summary.

Milestone 6.
Recommendation.

This milestone is a continuous result from previous milestone. We can see, the result of the prediction can be divided into three types. Figure 1 shows that the company is forecasted to decline. Thus, it is not a good choice to invest in such of this company. While figure 2 shows that the company only in stable situation. It is not going up nor down. This can be a choice for investor. However, the investor needs to reconsider if they intend to invest in this type of company for a long time period. Figure 3 is a very good type of investment because the company are predicted to go up.

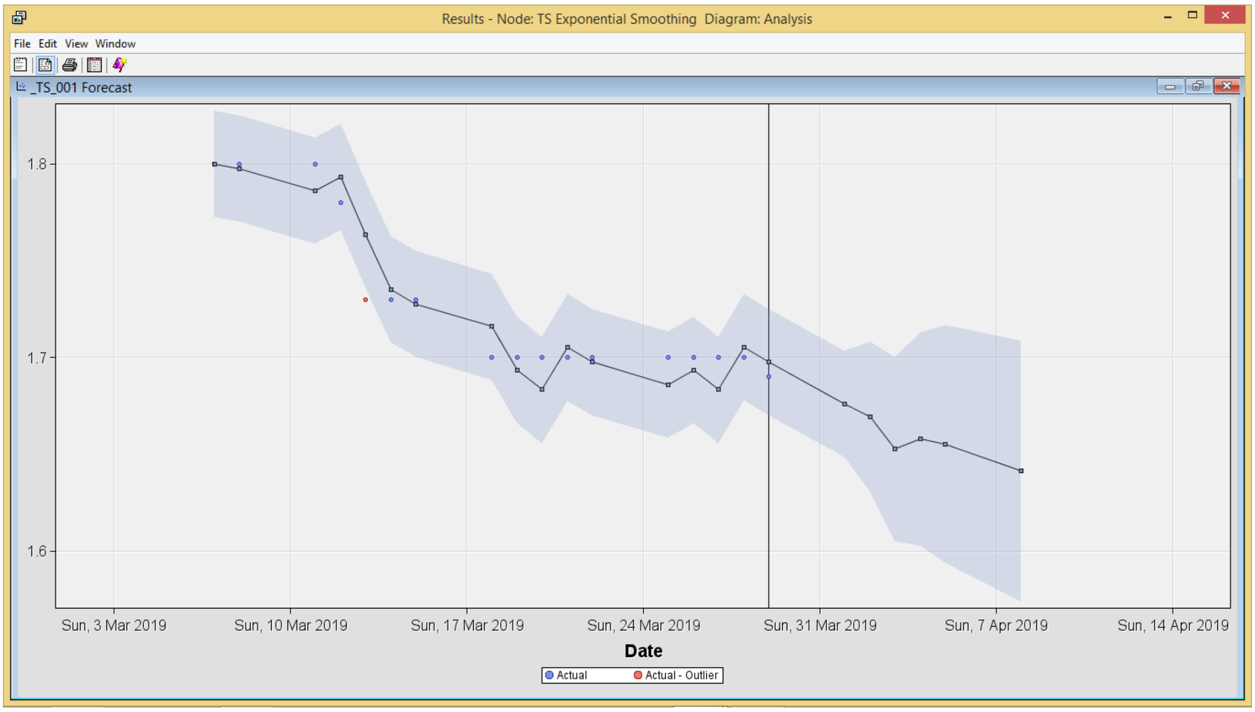


Figure 1: Type 1 company.

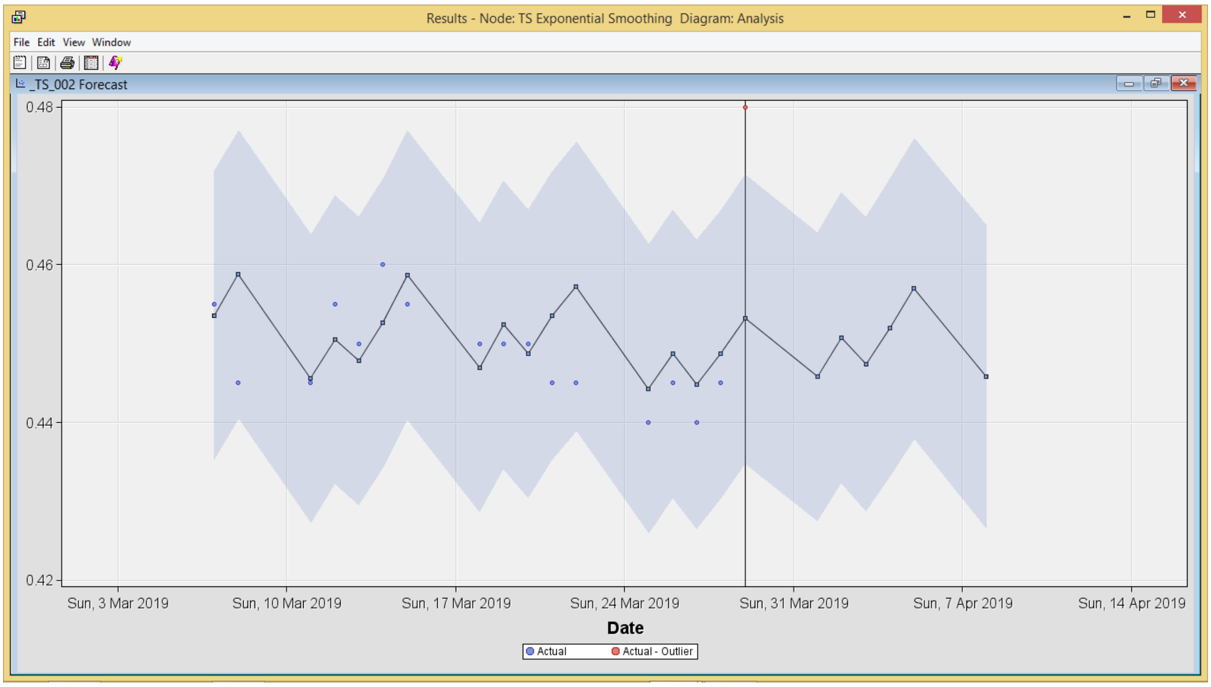


Figure 2: Type 2 Company.

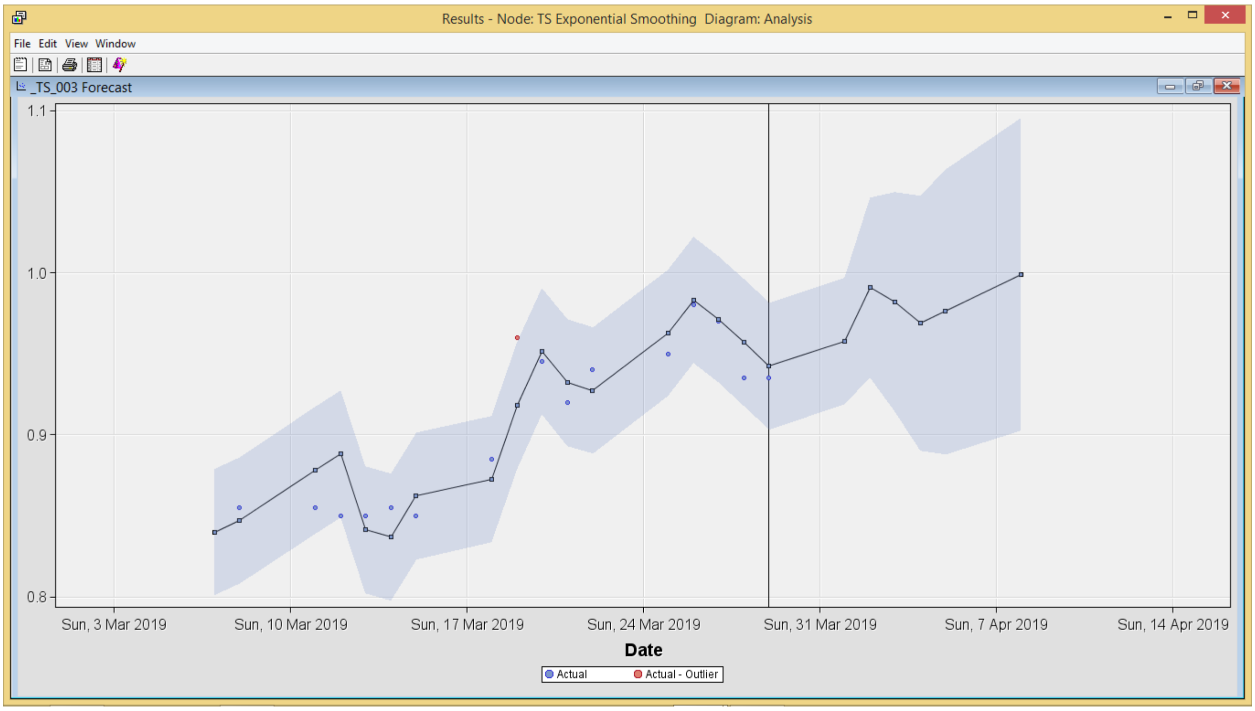


Figure 3: Type 3 Company.