WQD7005: Data Mining

Siti Salwa Binti Ab Rashid (WQD180060)

MILESTONE 1: Acquisition of Data

In order to achieve milestone 1, a python script has been developed to crawl data from stock market website in daily basis. This program is called as crawler and below is code snippet of the crawler.

pandas package is used to structures and do data analysis. KLSE.csv file is the input for the crawler to refer the company name list. Then, data frame is created and named as *companylist.*

```
import pandas as pd
df = pd.read_csv("KLSE.csv")
companylist = df["Name"].tolist()

Name = []
Code =[]
Open = []
High = []
Lowest = []
Last = []
Change = []
Volume = []
Buy = []
Sell = []
Date = []
Time = []

from lxml import html
import requests
```

AppCrawler class is created and page source is defined for each attribute that need to be acquired from stock market webpage.

```
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]
```

For the last part of script, webpage is defined and script will crawl each attributed needed based on company list obtained from the input file.

Webpage of daily stock market: https://www.thestar.com.my/business/marketwatch/stock-list/

Before the output is saved into csv file, a data frame is structured based on data acquire from the crawler.

```
for symbol in companylist:
    crawler = AppCrawler("https://www.thestar.com.my/business/marketwatch/stocks/?qcounter=" + symbol, 0)
    crawler.crawl()

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

# Store in a csv file
stock.to_csv('KLSE_030419_2pm.csv')
```

Snippet of data acquired by crawling from website.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Name	Code	Open Price	High Price	Low Price	Last Price	Change(%)	Volume	Buy Volume	Sell Volume	Date	Time
2	THREE-A RESOURCES BHD	12	0.845	0.84	0.84	0.845	0.6	400	0.845	0.85	Updated : 07 Mar 2019 |	1:06:00
3	ASTRAL ASIA BHD	7054	0.155	0.15	0.155	0.15	0	410	0.145	0.15	Updated : 07 Mar 2019 |	1:06:00
4	AIRASIA X BERHAD	5238	0.255	0.25	0.255	0.255	0	10	0.25	0.255	Updated : 07 Mar 2019 |	1:06:00
5	ABLEGROUP BERHAD	7086	0.07	0.07	0.07	0.07	0	460	0.07	0.075	Updated : 07 Mar 2019 |	1:06:00
6	ALLIANCE BANK MALAYSIA BERHAD	2488	4.21	4.19	4.2	4.19	-0.48	80	4.19	4.2	Updated : 07 Mar 2019 |	1:06:00
7	ACME HOLDINGS BERHAD	7131	0.25	0.25	0.25	0.25	8.7	220	0.25	0.3	Updated : 07 Mar 2019 |	1:06:00
8	ACOUSTECH BHD	7120	0.46	0.45	0.45	0.45	-1.1	13	0.45	0.455	Updated : 07 Mar 2019 |	1:06:00
9	ADVANCECON HOLDINGS BERHAD	5281	0.36	0.35	0.35	0.36	2.86	5	0.355	0.36	Updated : 07 Mar 2019 |	1:06:00
10	ADVENTA BHD	7191	0	0	0	0.355	0	0	0.355	0.385	Updated : 07 Mar 2019 |	1:06:00
11	ADVANCED PACKAGING TECHNOLOGY	9148	0	0	0	1.9	0	0	1.82	2	Updated : 07 Mar 2019 |	1:06:00
12	AE MULTI HOLDINGS BHD	7146	0.11	0.11	0.11	0.11	-8.33	1	0.11	0.115	Updated : 07 Mar 2019 |	1:06:00
13	AEON CO. (M) BHD	6599	1.61	1.58	1.59	1.6	0.63	3	1.6	1.61	Updated : 07 Mar 2019 |	1:06:00
14	AEON CREDIT SERVICE (M) BHD	5139	17.36	17.06	17.2	17.26	0.35	928	17.24	17.28	Updated : 07 Mar 2019 |	1:06:00
15	AFFIN BANK BERHAD	5185	2.31	2.28	2.29	2.29	-0.87	343	2.29	2.3	Updated : 07 Mar 2019 |	1:06:00
16	ABM FUJIYA BERHAD	5198	0	0	0	0.54	0	0	0.51	0.525	Updated : 07 Mar 2019 |	1:06:00
17	AHB HOLDINGS BHD	7315	0.13	0.13	0.13	0.13	0	500	0.13	0.135	Updated : 07 Mar 2019 |	1:06:00
18	APEX HEALTHCARE BHD	7090	9.67	9.53	9.66	9.53	-1.35	78	9.52	9.6	Updated : 07 Mar 2019 |	1:06:00

The crawler is executed 4 times in a day and it is continue for 2 weeks period.

1. First is between 9am-2pm.
2. Second is between 2pm-6pm.
3. Third is between 6pm-11pm.
4. Forth is after 11pm.

Each team member take turn to crawl data based on time period.

Other than daily stock market data, we also did crawl data for quarterly and annual. Python code snippet for quarterly and annual data crawler shown as below.

```
for 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)
```

Data crawled from website is by default in list structure, it was converted into data frame before save into excel format.

```
#import pandas to convert list to data frame
quarter=pd.DataFrame(QCode,columns=['Code'], dtype=str)
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_excel('Quarter Report.xlsx')

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

annual.to_excel('Annual Report.xlsx')
```

Snippet for each quarterly and annual data shown as below.

Quarterly:

	A	B	C	D	E	F	G	H	I	J	K	L
1		Code	EPS	DPS	NTA	Revenue	Profit/Loss	NQuarter	Quarter Date	Financial Date	Announced	Net
2	0	12	1.87	0	0.6679	120,354k	9,198k	4	31/12/2018	31/12/2018	20/2/2019	17.30%
3	1	12	1.71	2	0.6692	113,784k	8,398k	3	30/9/2018	31/12/2018	26/11/2018	23.80%
4	2	12	1.07	0	0.6521	101,361k	5,285k	2	30/6/2018	31/12/2018	7/8/2018	42.40%
5	3	12	1.27	0	0.6414	102,478k	6,238k	1	31/3/2018	31/12/2018	7/5/2018	39.60%
6	4	12	2.85	0	0.6287	109,423k	14,026k	4	31/12/2017	31/12/2017	20/2/2018	8.30%
7	5	12	1.65	1.8	0.6241	96,542k	8,125k	3	30/9/2017	31/12/2017	6/11/2017	19.90%
8	6	12	2.12	0	0.6915	102,338k	9,174k	2	30/6/2017	31/12/2017	17/8/2017	0.40%
9	7	12	2.62	0	0.7361	103,182k	10,323k	1	31/3/2017	31/12/2017	11/5/2017	54.30%
10	8	12	3.29	0	70.99	95,037k	12,953k	4	31/12/2016	31/12/2016	23/2/2017	322.10%
11	9	12	2.58	1.8	0.6942	88,226k	10,138k	3	30/9/2016	31/12/2016	15/11/2016	55.90%
12	10	12	2.32	0	0.6679	96,887k	9,138k	2	30/6/2016	31/12/2016	19/8/2016	30.60%
13	11	12	1.7	0	0.6446	107,568k	6,692k	1	31/3/2016	31/12/2016	5/5/2016	90.40%
14	12	12	0.78	0	0.6305	93,924k	3,071k	4	31/12/2015	31/12/2015	24/2/2016	35.10%
15	13	12	1.65	1.4	0.6386	91,134k	6,503k	3	30/9/2015	31/12/2015	24/11/2015	38.90%
16	14	12	1.78	0	0.6163	92,749k	6,995k	2	30/6/2015	31/12/2015	14/8/2015	34.40%
17	15	12	0.89	0	0.5985	74,593k	3,515k	1	31/3/2015	31/12/2015	5/5/2015	2.40%
18	16	12	1.18	0	0.5888	77,299k	4,646k	4	31/12/2014	31/12/2014	16/2/2015	38.20%
19	17	12	1.19	0	0.5903	72,963k	4,681k	3	30/9/2014	31/12/2014	14/11/2014	163.40%
20	18	12	1.32	0	0.5779	84,510k	5,203k	2	30/6/2014	31/12/2014	14/8/2014	112.60%

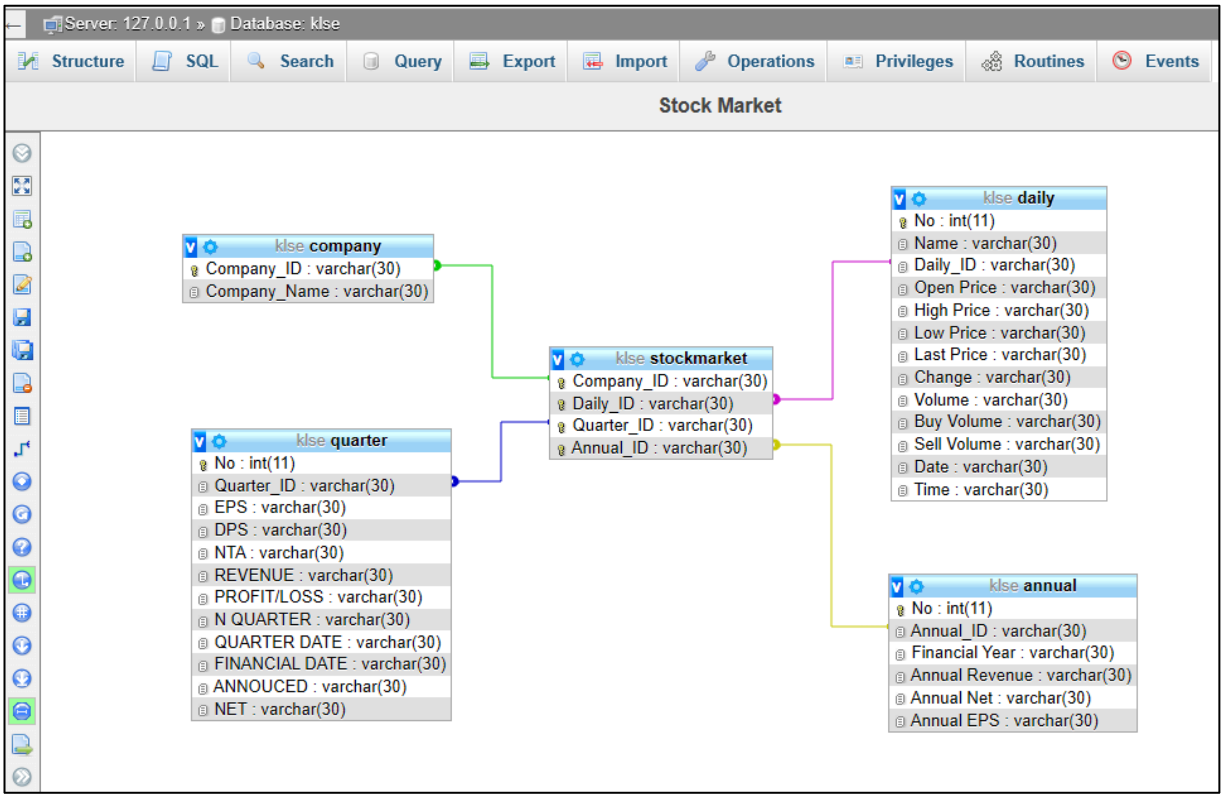
Annual:

	A	B	C	D	E	F
1		Code	Financial Year	Annual Revenue	Annual Net	Annual EPS
2	0	12	31-Dec-18	437,977	29,119	5.92
3	1	12	31-Dec-17	411,485	41,648	9.24
4	2	12	31-Dec-16	387,718	38,921	9.89
5	3	12	31-Dec-15	352,400	20,084	5.1
6	4	12	31-Dec-14	311,410	18,130	4.6
7	5	12	31-Dec-13	302,910	10,182	2.58
8	6	12	31-Dec-12	306,428	17,532	4.46
9	7	12	31-Dec-11	268,806	15,886	4.04
10	8	7054	31-Dec-18	25,728	-5,351	-0.81
11	9	7054	31-Dec-17	31,489	-1,647	-0.25
12	10	7054	31-Dec-16	25,813	5	-1.54
13	11	7054	31-Dec-15	24,583	-5,408	-4.51
14	12	7054	31-Dec-14	28,849	-776	-0.65
15	13	7054	31-Dec-13	32,324	2,221	1.85
16	14	7054	31-Dec-12	36,855	3,475	2.9
17	15	7054	31-Dec-11	38,497	7,588	6.32
18	16	5238	31-Dec-18	4,544,450	-312,697	-7.6
19	17	5238	31-Dec-17	4,562,005	98,886	2.3
20	18	5238	31-Dec-16	4,006,534	230,539	5.5

MILESTONE 2: Management of Data

After data acquired and crawled for 2 weeks, now is time to store and manage the collected data.

We have create our own database to store and manage the crawled data by implementing *Star Schema* model.



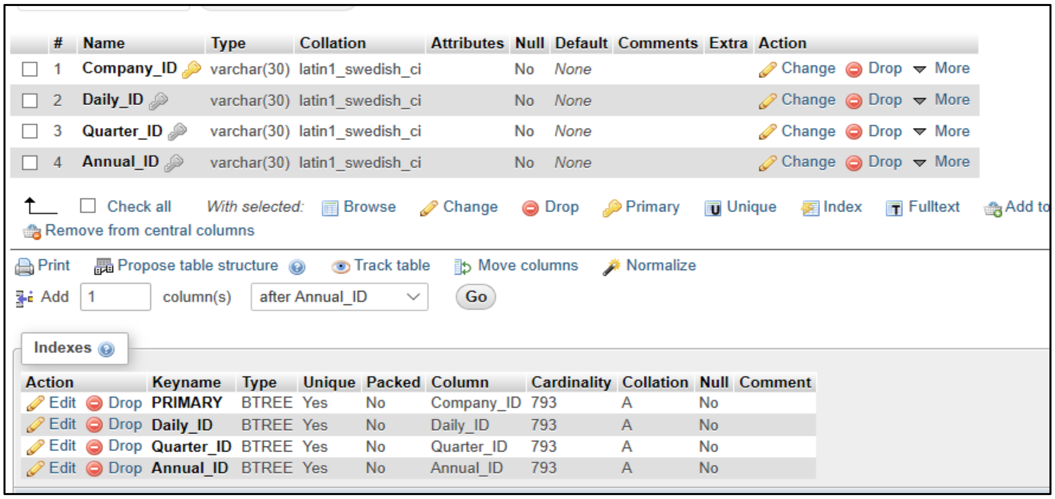
From our *Star Schema*, there are 4 *dimension tables* connected to *fact table* in the middle. Fact table contains all unique ID for each dimension table where we create dimension table based on company details, daily data, quarterly data and also annual data that we obtained from milestone 1. Dimension table contain detail information for each dataset.

Above star schema which also known as relational database is part of dimensional model and model can also be instantiated in as multidimensional database, known as OLAP (Online analytical processing). Typically OLAP operation can perform action such as:

1. Roll up (drill up) – to summarize data by climbing up hierarchy or by dimension reduction.
2. Drill down (roll down) – to reverse of roll up from higher level summary to lower level summary or detailed data, or introducing new dimensions.
3. Slice and dice – project and select certain data.

Hence, by creating our data warehouse model like this, it is easier to access the database through drill down and roll up method.

Below is table structure for our *stockmarket* database.



#	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

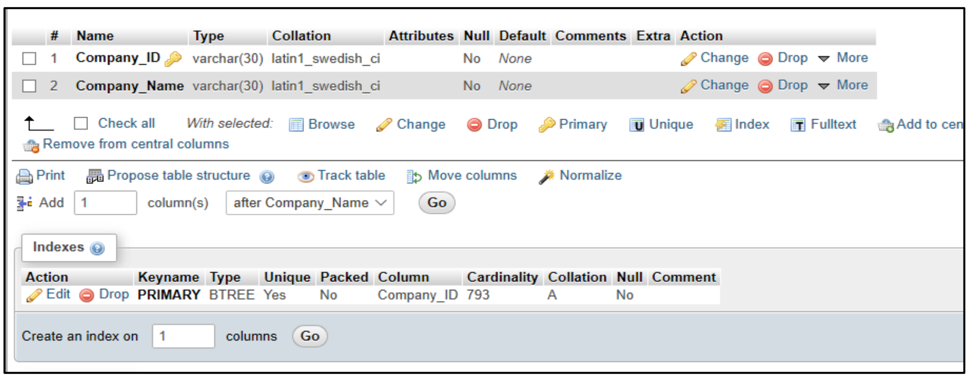
Check all With selected: Browse Change Drop Primary Unique Index Fulltext Add to
Remove from central columns

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	

Table structure for *company* table:



#	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

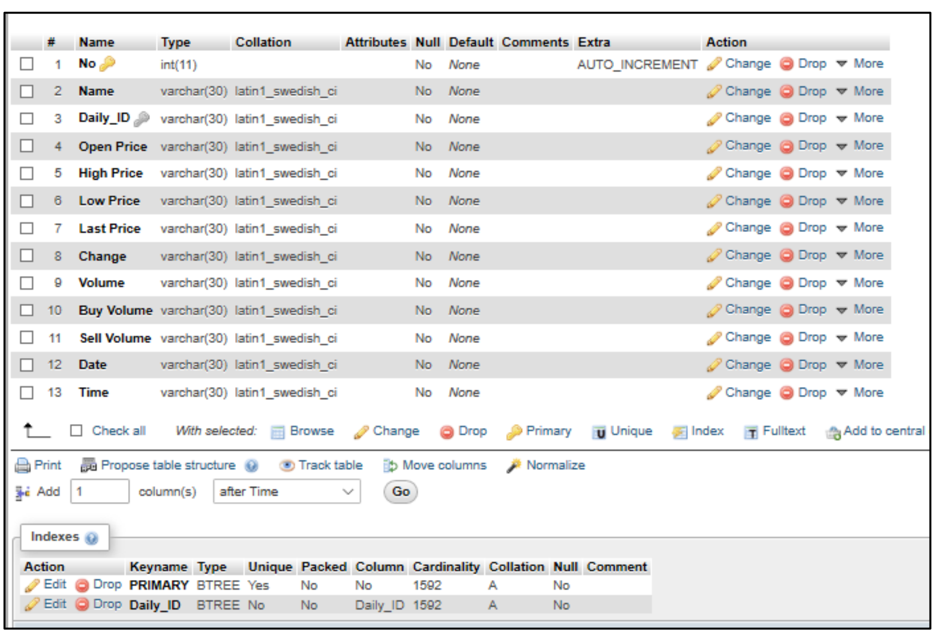
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after Company_Name Go

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

Table structure for *daily* table:



#	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	

Table structure for *quarter* table:

Table structure | Relation view

#	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	

Table structure for *annual* table:

Table structure | Relation view

#	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

Check all *With selected:* Browse Change Drop Primary Unique Index Fulltext Add to central
Remove from central columns

Print Propose table structure Track table Move columns Normalize

Add 1 column(s) after Annual EPS Go

Indexes

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	

For data query purposes, we have set up our Hive in Hadoop architecture. Since our dataset is structured data, hence using Hive is one of good option we have to manage and query it. Below is steps on how we create table in Hive by using Hadoop Distributed File System (HDFS) approach.

Firstly, create *klse* directory and 3 other subdirectories under *klse* to place 3 tables of *daily, quterly and annual* data.

```
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
```

Next, upload the datasets into HDFS based on respective 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
```

Create Hive table, import datasets and verify the data is ready by issue SQL command. This steps is repeated for all 3 datasets.

```
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>
```

```
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>
```

```
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>
```

Data that placed in each table is now ready to be query by simply using Hive SQL command.

MILESTONE 3: Processing of Data

When it comes to processing data part, we used Python to perform this task.

For pre-processing part, we manage the missing value and prepare the data by using Pyhton. Below code used for preprocess our data.

Initial step, we declare package that will be used; *pandas, numpy and glob.* Then we create data frame, merge it into a single data frame then initiate pre-processing by checking the data type.

```
import pandas as pd
import numpy as np
import glob

###############################################################
# Data Processing for Stock Market------------------------
# Create list of file paths from a directory
paths = []

for filepath in glob.iglob('D:/Web Crawler/Klse Data/*'):
    paths.append(filepath)

#create list of dataframes using file paths
df_list = []
for file in paths:
   df_list.append(pd.read_excel(file))

#Merge a list of dataframe into one dataframe
stock_data = pd.concat(df_list)

#Check data types
print (stock_data.dtypes)
```

For daily data, below steps consider for pre-processing:

1. Change *Code* attribute into string for easier analysis and plotting purposes.
2. Strip Unwanted Character in Column Date.
3. Convert String to Date format.
4. Convert Certain String Column to Numeric.
5. Replace missing value with 0.
6. Delete column *Time* as we will not use this in our analysis.
7. Drop duplicate observation in a data frame.
8. Add a *Class* Column for categorize each company stock market pattern.

Then, the clean and preprocessed data is saved in excel format.

```
#Change code into string
stock_data['Code']=stock_data['Code'].apply(lambda x: '{0:0>4}'.format(x))

#Strip Unwanted Character in Column Date
stock_data['Date'] = stock_data['Date'].map(lambda x: x.lstrip('Updated : ').rstrip(' |'))

#Convert String to Date format
stock_data['Date'] = pd.to_datetime(stock_data['Date'], format = '%d %b %Y')

#Convert Certain String Column to Numeric
stock_data['Open Price'] = pd.to_numeric(stock_data['Open Price'],errors='coerce')
stock_data['High Price'] = pd.to_numeric(stock_data['High Price'],errors='coerce')
stock_data['Low Price'] = pd.to_numeric(stock_data['Low Price'],errors='coerce')
stock_data['Last Price'] = pd.to_numeric(stock_data['Last Price'],errors='coerce')
stock_data['Change (%)'] = pd.to_numeric(stock_data['Change (%)'],errors='coerce')
stock_data['Volume'] = pd.to_numeric(stock_data['Volume'],errors='coerce')

#replace missing value with 0
stock_data = stock_data.replace(np.nan, 0, regex=True)

#Delete column 'Time'
del stock_data['Time']

#Drop duplicate observation in a dataframe
stock_data = stock_data.drop_duplicates(keep = False)

#Add a 'Class' Column
stock_data['Class'] = 'Constant'
stock_data.loc[stock_data['Change (%)'] > 0, 'Class'] = 'Up'
stock_data.loc[stock_data['Change (%)'] < 0, 'Class'] = 'Down'

#Save as excel file
stock_data.to_excel('Clean Stock Market Data.xlsx')
```

The necessary steps as done for daily data is repeated for quarterly and annual data as well.

Data pre-processing for quarterly data:

```
#Data Processing for Quarter Report---------------------------------------------------
quarter = pd.read_excel("Quarter Report.xlsx",
                     sheet_name = 0,
                     header = 0,
                     index_col = False,
                     keep_default_na = True)

#Convert Code to string by adding leading zero
quarter['Code']=quarter['Code'].apply(lambda x: '{0:0>4}'.format(x))

#Convert String to Date format
quarter['Financial Year'] = pd.to_datetime(quarter['Financial Year'], format = '%d %b %Y')

#Delete column 'No' and 'Financial Date'
del quarter['No']
del quarter['Financial Date']
del quarter['Announced']

#Check data types
print(quarter.dtypes)

#Strip Unwanted Character in Column Revenue and Profit/Loss
quarter['Revenue'] = quarter['Revenue'].str.replace('k','')
quarter['Revenue'] = quarter['Revenue'].str.replace(',','')
quarter['Revenue'] = quarter['Revenue'] + "000"
quarter['Profit/Loss'] = quarter['Profit/Loss'].str.replace('k','')
quarter['Profit/Loss'] = quarter['Profit/Loss'].str.replace(',','')
quarter['Profit/Loss'] = quarter['Profit/Loss'] + "000"

# Change Profit/loss and Revenue to numeric
quarter['Revenue'] = pd.to_numeric(quarter['Revenue'], errors ='coerce')
quarter['Profit/Loss'] = pd.to_numeric(quarter['Profit/Loss'], errors ='coerce')

#Drop duplicate observation in a dataframe
quarter = quarter.drop_duplicates(keep = False)

quarter.to_excel("Quarter Report.xlsx")
```

Data pre-processing for annual data:

```
######################################################################################
# Data Processing for Annual Report-------------------------------------------------------
annual = pd.read_excel("Annual Report.xlsx",
                     sheet_name = 0,
                     header = 0,
                     index_col = False,
                     keep_default_na = True)

#Convert Code to string by adding leading zero
annual['Code']=annual['Code'].apply(lambda x: '{0:0>4}'.format(x))

#Convert String to Date format
annual['Financial Year'] = pd.to_datetime(annual['Financial Year'], format = '%d %b %Y')

#Delete column 'No' and 'Financial Date'
del annual['No']
del annual['Financial Date']

#Drop duplicate observation in a dataframe
annual = annual.drop_duplicates(keep = False)

annual.to_excel("Annual Report.xlsx")
```

Next, for data reduction and feature selection we implement Piecewise Aggregate Approximation (PAA) and Symbolic Aggregate Approximation (SAX) technique by sliding window size of data and help to determine the best feature to be used for further analysis. Below snippets show the Python code used for this implementation.

Initial part is import packages and declare technique to be used and also dataset.

```
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_SymbolicAggregateApproximation

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

Implementation of PAA and SAX.

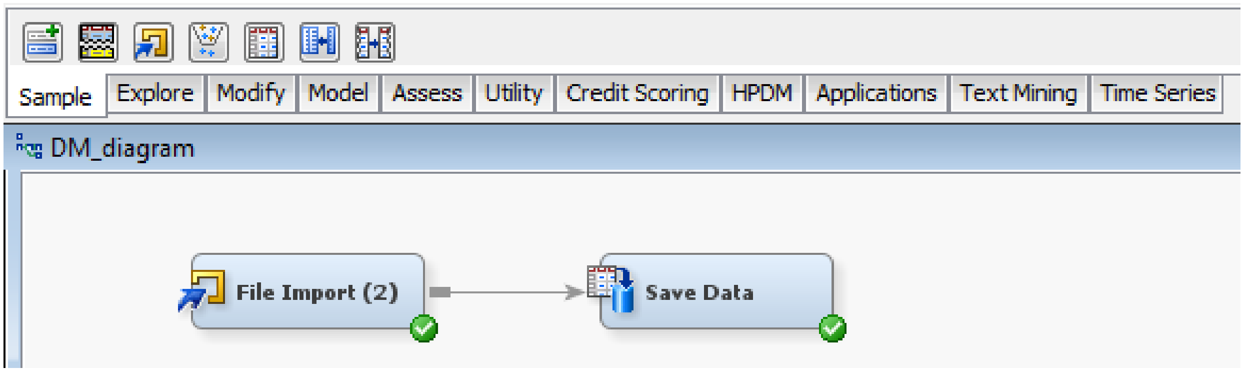
```
# PAA transform (and inverse transform) of the data
n_paa_segments = 10
paa = PiecewiseAggregateApproximation(n_segments=n_paa_segments)
paa_dataset_inv = paa.inverse_transform(paa.fit_transform(dataset))

# 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_sax_symbols_avg,
                                                alphabet_size_slope=n_sax_symbols_slope)
one_d_sax_dataset_inv = one_d_sax.inverse_transform(one_d_sax.fit_transform(dataset))
```

Visualize the output.

```
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()
```

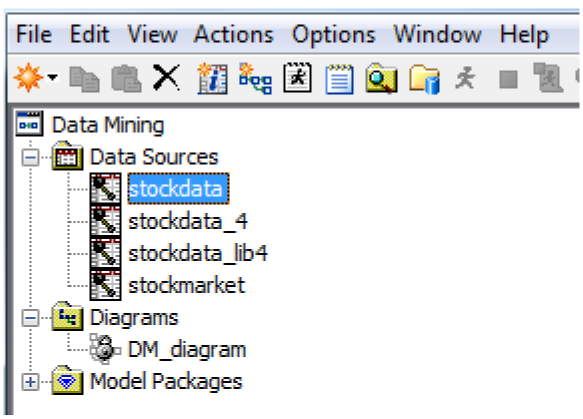
MILESTONE 4: Interpretation of Data

For milestone 4 where we reach part interpreting data, SAS Enterprise Miner is used as per requirement. Prior to perform any analysis in SAS Enterprise Miner, dataset has been imported and stored in project directory as *Data Source.*

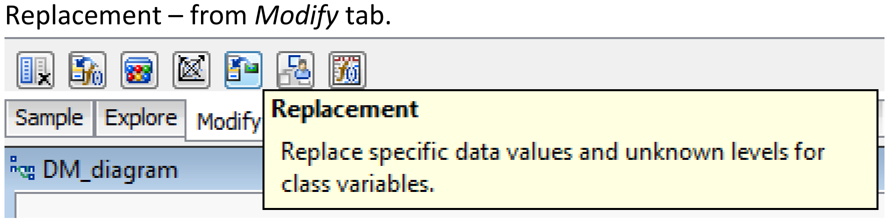


Next, few nodes were used to create model and visualize the output for analysis purposes. The nodes involved were:

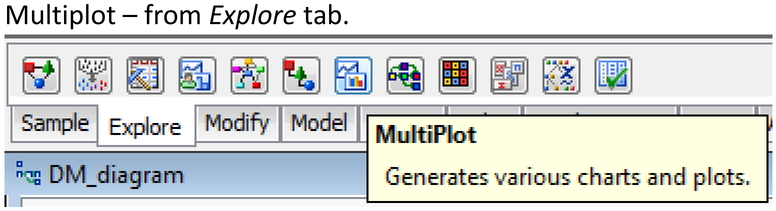
1. stockdata – from *Data Sources*



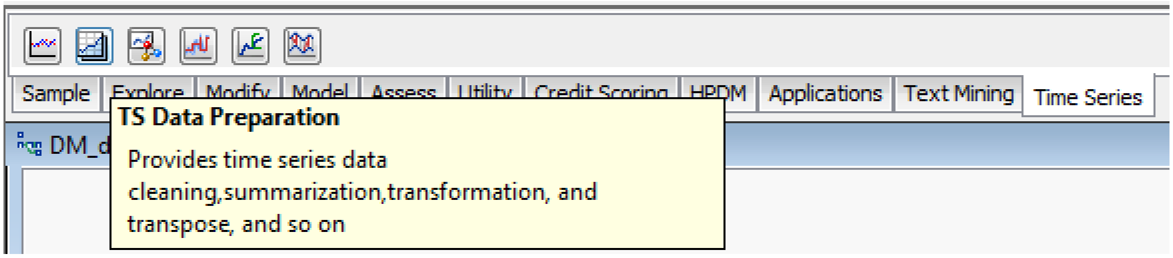
2. Replacement – from *Modify* tab.



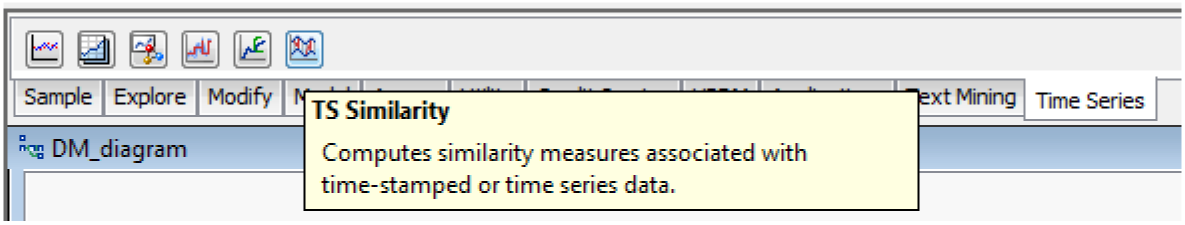
3. Multiplot – from *Explore* tab.



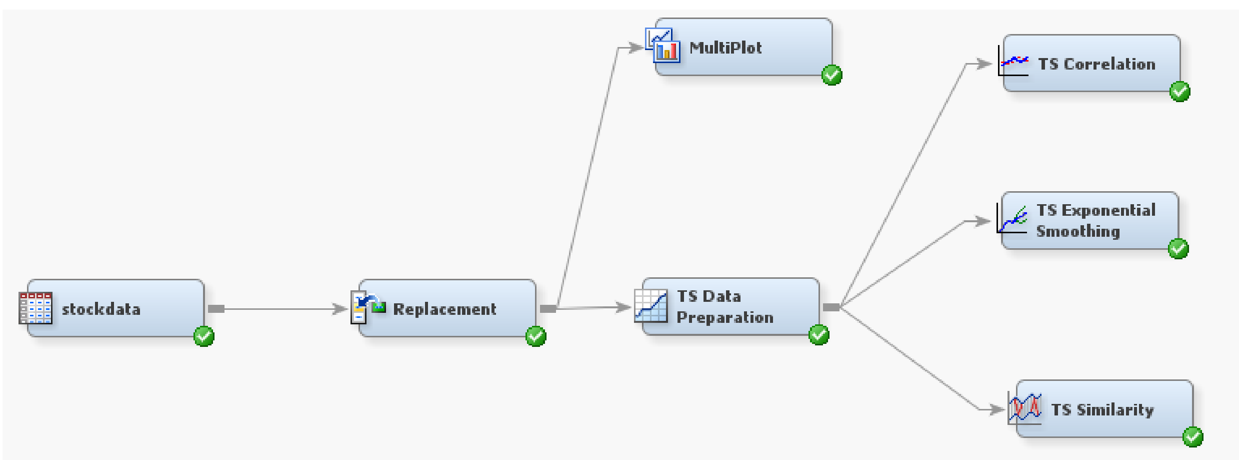
4. TS Data Preparation – from *Time Series* tab.



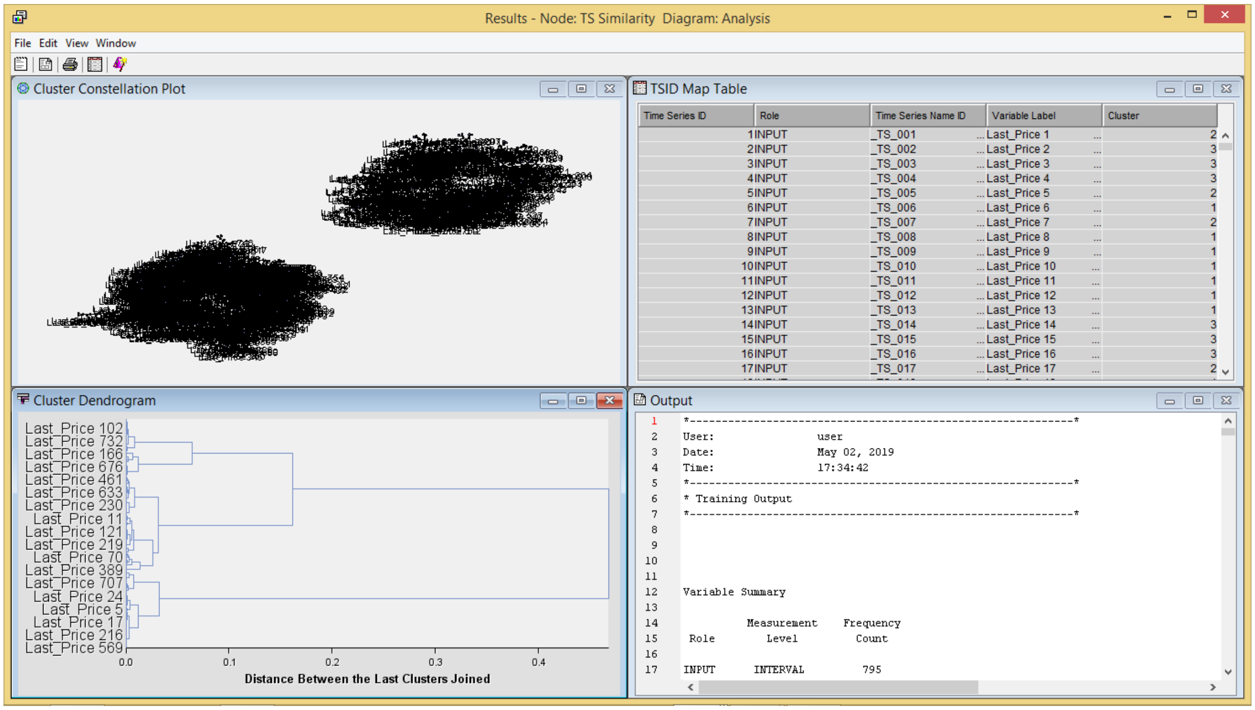
5. TS Similarity – from *Time Series* tab.



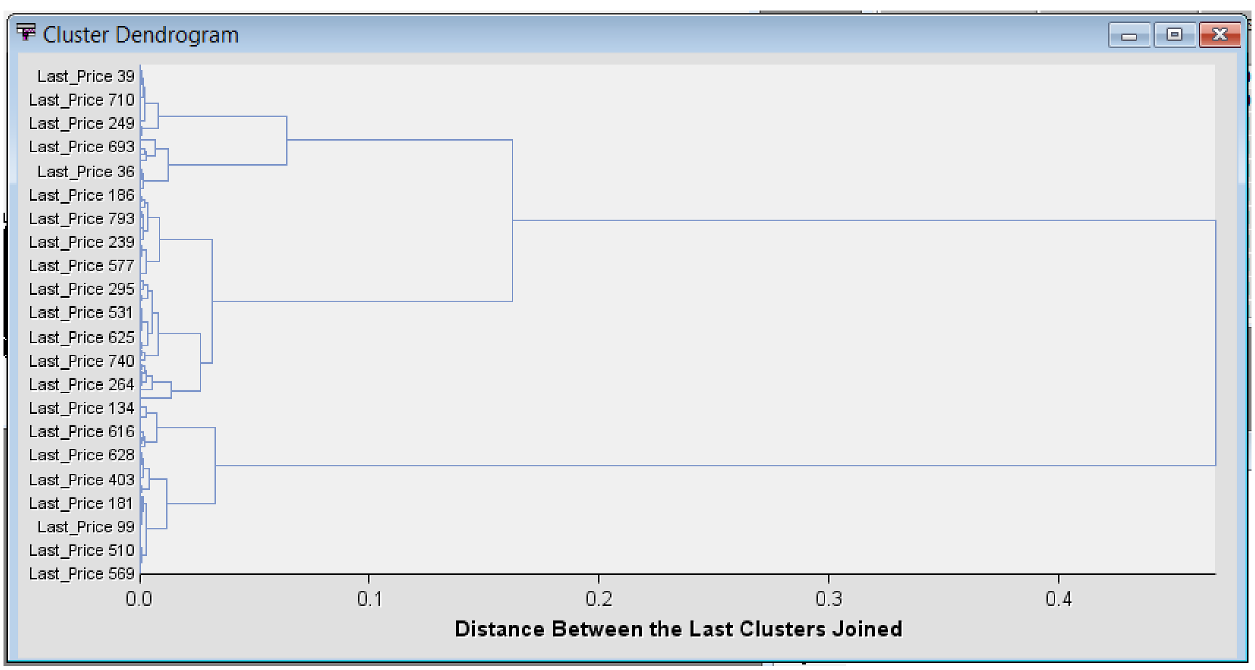
Below is overall diagram for the workflow.



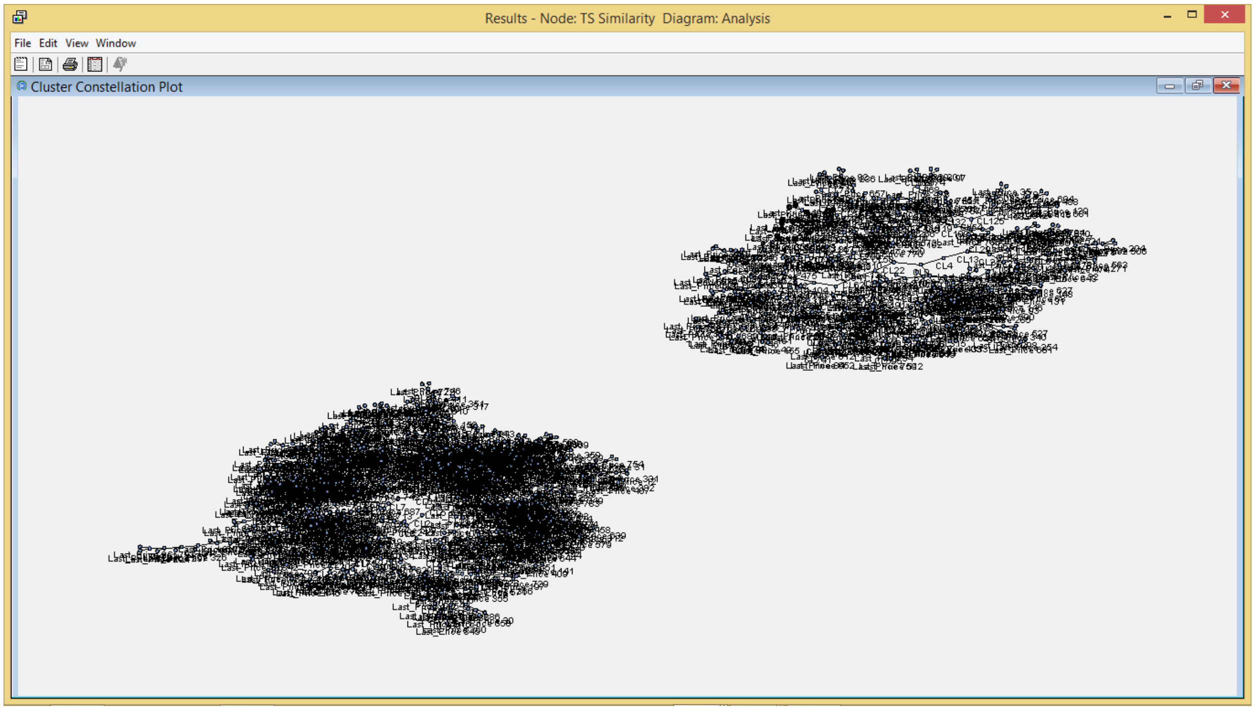
Based on result shown by TS Similarity node, the stock market data can be clustered into 3 clusters as below.



Clustering method used in this analysis is by using Hierarchical Clustering method and the number of cluster is determine by plotting the dendogram. It shown that the optimum number of cluster is 3.



Although cluster plot shows like there are only 2 clusters, but the actual is there are 3 clusters.



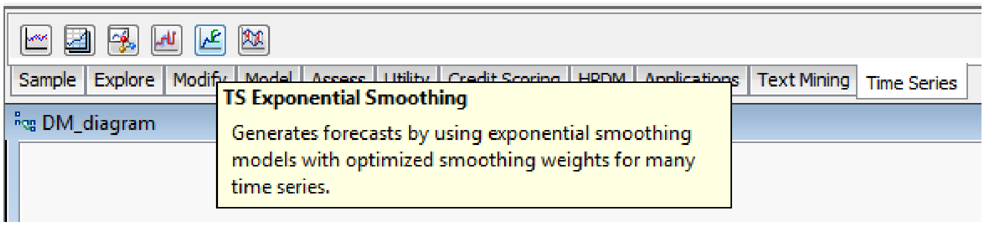
Details for clustering can be seen from TSID map table as well which show clear distribution of data into 3 clusters.

TSID Map Table

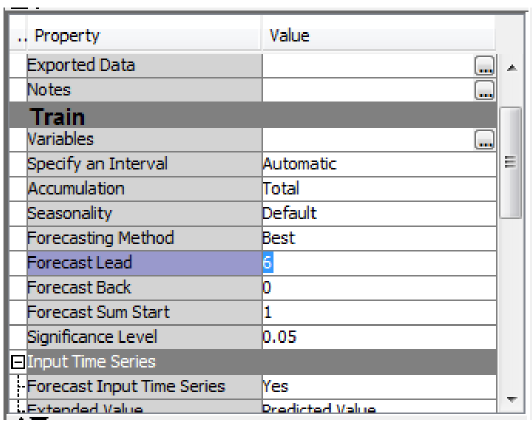
Time Series ID	Role	Time Series Name ID	Variable Label ▲	Cluster
1	INPUT	_TS_001	Last_Price 1 ...	2
10	INPUT	_TS_010	Last_Price 10 ...	1
100	INPUT	_TS_100	Last_Price 100 ...	3
101	INPUT	_TS_101	Last_Price 101 ...	1
102	INPUT	_TS_102	Last_Price 102 ...	3
103	INPUT	_TS_103	Last_Price 103 ...	2
104	INPUT	_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

MILESTONE 5: Communication of Insights

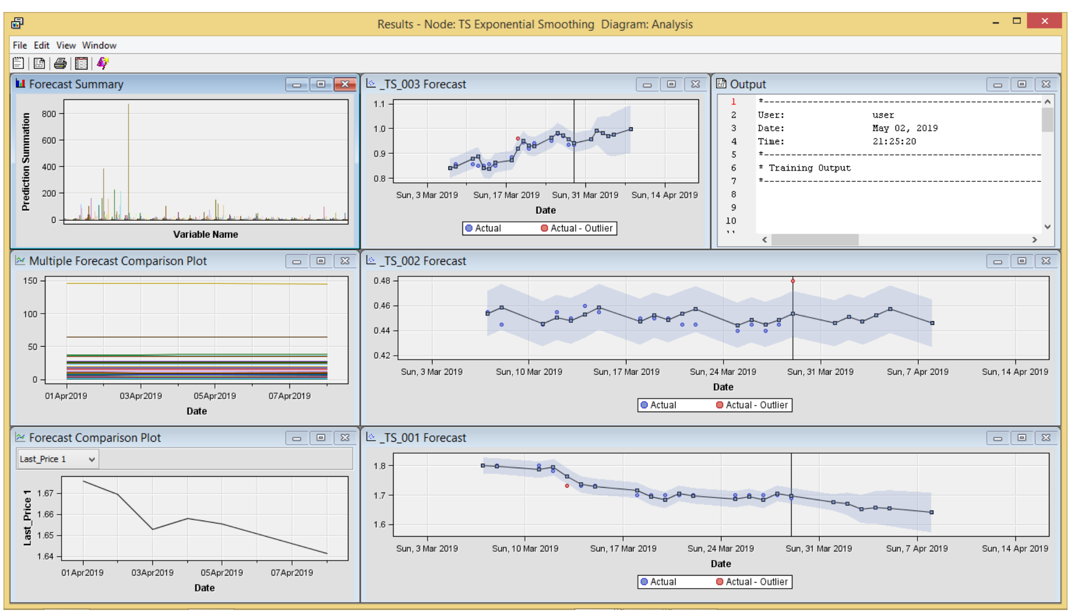
To dig further about the data, I used other nodes as well in SAS Enterprise Miner such as TS Exponential Smoothing from *Time Series* tab for forecasting purposes.



The forecasting points has been set to 6 points which is I think enough for data that we crawled for 2 weeks period.



Below shown general output produced by TS Exponential Smoothing node.



I highlighted 3 companies that shown different behavior based on output graph, they are _TS_003, _TS_002 and _TS_001. To identify which company is represent by this time series ID, TSID Map Table is referred. This table can be found from result shown by TS Data Preparation node.

Results - Node: TS Data Preparation (2) Diagram: Analysis

File Edit View Window

TSID Map Table

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

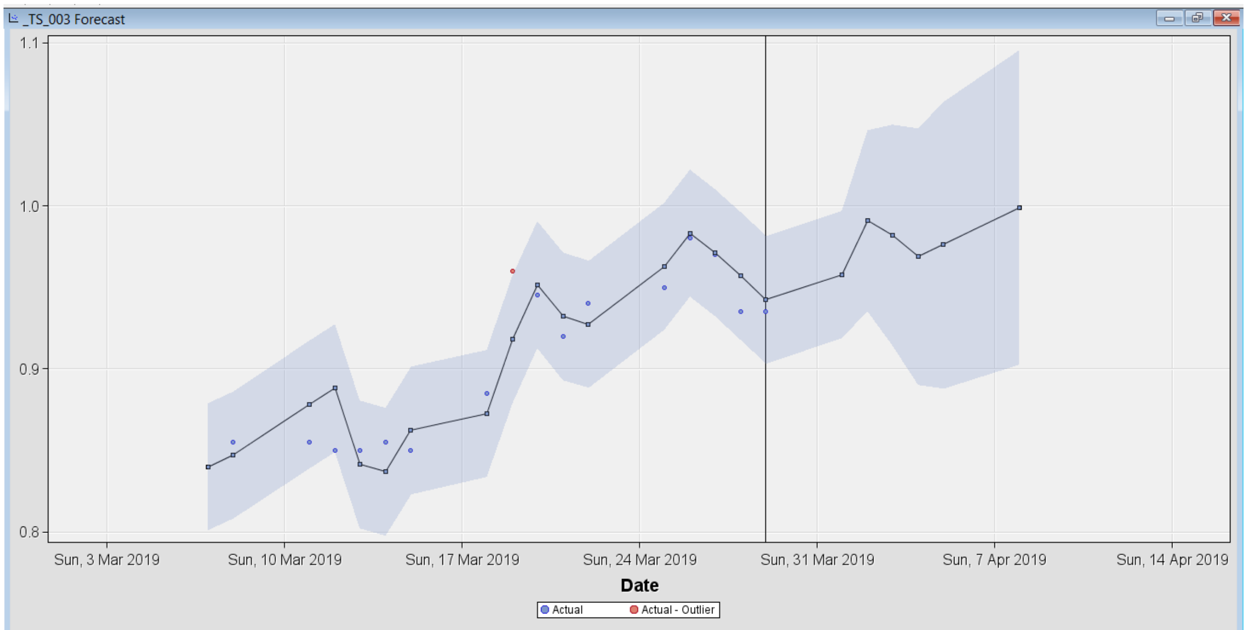
From above mapping table, we can see that the three plots is represent by company code 0012, 0008 and 0002 respectively. I have simplify the details as per table below.

TSID (Time Series ID)	Company Code	Company Name
_TS_003	0012	THREE-A RESOURCES BHD
_TS_002	0008	WILLOWGLEN MSC BHD
_TS_001	0002	KOTRA INDUSTRIES BHD

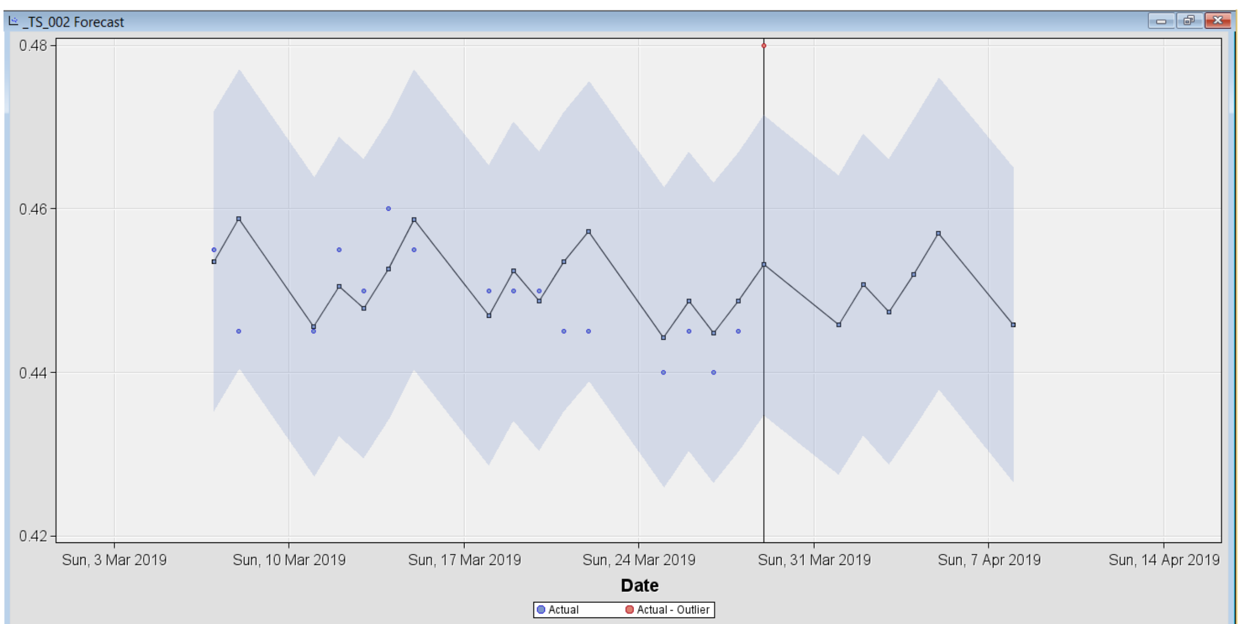
Details of company name can be found from input file that we used in milestone 1. Below is screenshot from the input file that showing company code and company name for 3 highlighted time series above.

	A	B	C	D	E	F	G	H	I	J	K
1	Name	Code	Open Price	High Price	Low Price	Last Price	Change	Volume	Buy Volume	Sell Volume	Date
2	THREE-A RESOURCES BHD	0012	0.845	0.84	0.84	0.84	0	485	0.840 / 485	0.850 / 315	3/7/2019
767	WILLOWGLEN MSC BHD	0008	0.455	0.45	0.45	0.455	1.11	1210	0.455 / 589	0.460 / 100	3/7/2019
1173	KOTRA INDUSTRIES BHD	0002	0	0	0	1.8	0	0	1.750 / 50	1.800 / 48	3/8/2019

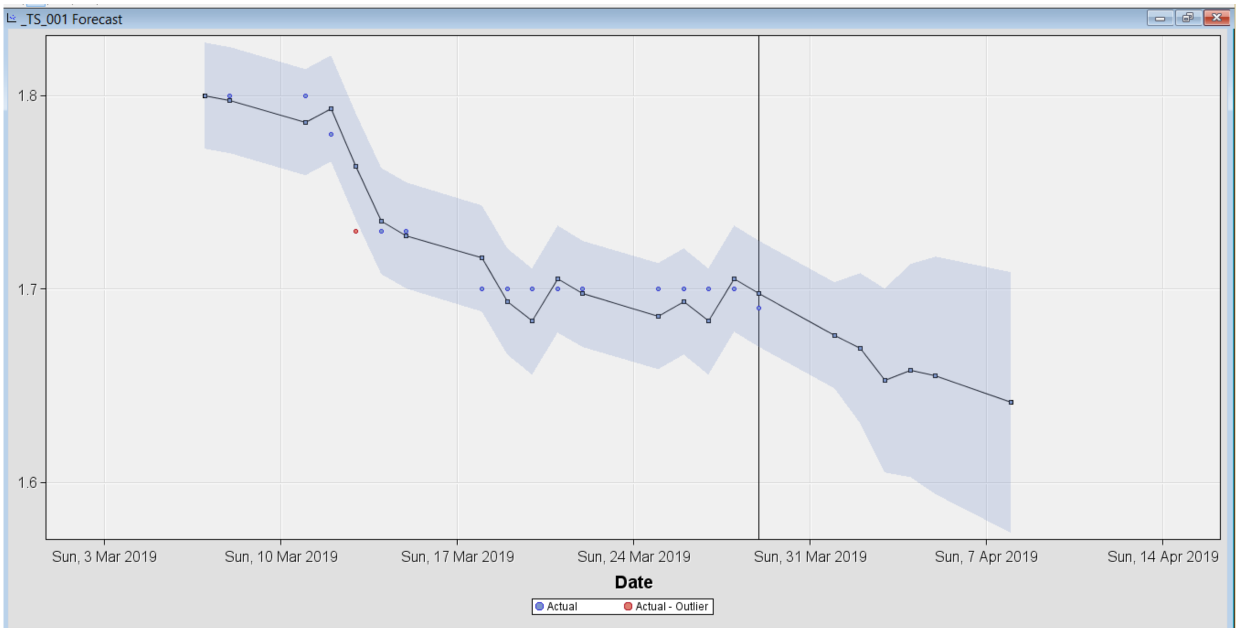
Time series plot for company THREE-A RESOURCES BHD which represent by graph TS_003 shows consistent increment for the next 6 days. Although there are 2 data points which show the decrement, but in general the graph demonstrate a stable increment for total 3 weeks period (actual and forecast).



While for company WILLOWGLEN MSC BHD, time series plot shows that there is constant moving of forecasting in general. Looking at the pattern for 2 weeks period, the price is never go beyond its support price value which is within 0.44 and 0.46.



The third company that I wanted to highlight is KOTRA INDUSTRIES BHD which obviously shown a drastic decrement for its stock market price. There is only 1 forecast data point increase, then it constantly goes down.



MILESTONE 6: Recommendation

Based on outcome gained from milestone 1 until milestone 5, I have come out with summary table as below where as mentioned in previous part, I just focus on three company where they show different behavior in their stock market price pattern and forecasting.

TSID (Time Series ID)	Company Code	Company Name	Stock Price Pattern	Field of Business
_TS_003	0012	THREE-A RESOURCES BHD	Increase	Food Industries
_TS_002	0008	WILLOWGLEN MSC BHD	Constant	Technology Industries
_TS_001	0002	KOTRA INDUSTRIES BHD	Decrease	Pharmaceutical Industries

From above summary table I can conclude and recommend that investor should continue investing in food industries since there is always high demand in this area. Food and beverages (F&B) industries also one of business that provide high profit return and make it suitable for long term investment.

For investor who wish to invest in technology industries, it is advised to put their investment for a short term period. For instance, investor first can identify what is support price for company that they interested to invest, then can buy stock during the lowest price and sell it when it reach the highest price based on its support price.

On the other hand, investing in pharmaceutical industries is quite risky nowadays. The stock price pattern shows radically decrease within this 2 weeks period analysis. This might be caused by there are external factors that impacting this industries such as imported raw material price and currency rate between countries.