UNIVERSITY OF MALAYA

EXAMINATION FOR THE DEGREE OF MASTER OF DATA SCIENCE

ACADEMIC SESSION 2019/2020 : SEMESTER II

WQD7005 : Data Mining

Name of Candidate: Lew Teck Wei

Matric Number: 17021682

June 2020

INSTRUCTIONS TO CANDIDATES :

Answer **ALL** questions (50 marks).

(This question paper consists of 5 questions on 3 printed pages)

Mini-assignment (50 marks)

Instructions: Work individually, submission via Spectrum.

. 1. You are required to make a user-agent that will crawl the WWW (your familiar domain) to produce dataset of a particular website.
   - the web site can be as simple as a list of webpages and what other pages they link to
   - the output does not need to be in XHTML (or HTML) form
     a multi-stage approach (e.g. produce the xhtml or html in csv format)

Answer:
Scrape the following date from WWW using the library and code as below:

➔ refer to video presentation https://youtu.be/lqMMceTyH5A for better understanding or refer to the presentation slide at github https://github.com/L-venLewTeckWei/Final-Exam-Q1-and-Q2-Lew-Teck-Wei-17021682

MBB

	date	close
0	2020-03-09	8.239999771118164
1	2020-03-06	8.5
2	2020-03-05	8.529999732971191
3	2020-03-04	8.470000267028809
4	2020-03-03	8.40999984741211
...	...	...
1237	2015-03-16	9.09000015258789
1238	2015-03-13	9.100000381469727
1239	2015-03-12	9.119999885559082
1240	2015-03-11	9.09000015258789
1241	2015-03-10	9.229999542236328

[1242 rows x 2 columns]

CIMB

	date	close
0	2020-03-09	4.199999809265137
1	2020-03-06	4.440000057220459
2	2020-03-05	4.5
3	2020-03-04	4.539999961853027
4	2020-03-03	4.5
...	...	...
1238	2015-03-16	5.909999847412109
1239	2015-03-13	5.920000076293945
1240	2015-03-12	5.849999904632568
1241	2015-03-11	5.800000190734863
1242	2015-03-10	5.949999809265137

[1243 rows x 2 columns]

RHB

	date	close
0	2020-03-09	5.46999979019165
1	2020-03-06	5.699999809265137
2	2020-03-05	5.710000038146973
3	2020-03-04	5.710000038146973
4	2020-03-03	5.639999866485596
...	...	...
1238	2015-03-16	7.265110015869141
1239	2015-03-13	7.199659824371338
1240	2015-03-12	7.237060070037842
1241	2015-03-11	7.349259853363037
1242	2015-03-10	7.386660099029541

[1243 rows x 2 columns]

KLCI

	date	close
0	2020-03-09	1424.1600341796875
1	2020-03-06	1483.0999755859375
2	2020-03-05	1491.030029296875
3	2020-03-04	1489.949951171875
4	2020-03-03	1478.6400146484375
...	...	...
1221	2015-03-16	1780.5400390625
1222	2015-03-13	1781.75
1223	2015-03-12	1786.8699951171875
1224	2015-03-11	1778.1600341796875
1225	2015-03-10	1789.7299804687 5

[1226 rows x 2 columns]

DJI

	date	close
0	2020-03-09	23851.01953125
1	2020-03-06	25864.779296875
2	2020-03-05	26121.279296875
3	2020-03-04	27090.859375
4	2020-03-03	25917.41015625
...	...	...
1254	2015-03-16	17977.419921875
1255	2015-03-13	17749.310546875
1256	2015-03-12	17895.220703125
1257	2015-03-11	17635.390625
1258	2015-03-10	17662.939453125

[1259 rows x 2 columns]

SNP

	date	close
0	2020-03-09	2746.56005859375
1	2020-03-06	2972.3701171875
2	2020-03-05	3023.93994140625
3	2020-03-04	3130.1201171875
4	2020-03-03	3003.3701171875
...	...	...
1254	2015-03-16	2081.18994140625
1255	2015-03-13	2053.39990234375
1256	2015-03-12	2065.949951171875
1257	2015-03-11	2040.239990234375
1258	2015-03-10	2044.1600341796875

[1259 rows x 2 columns]

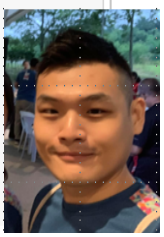
Step 1 : Data Scraping – crawl the trading date and closing price of the following Index and Stocks

- Maybank –MBB
- RHB
- KLCI – Kuala Lumpur Composite Index
- DJI – Dow Jone Index
- SNP - S&P 500 Index

```
import requests
import bs4
import json
import datetime
import pandas as pd
page = requests.get("https://finance.yahoo.com/quote/MBBM.KL/history?period1=1425945600&period2=1583798400&interval=1d&filter

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

listScript = soup.find_all("script")
for script in listScript:
    txtScript = script.string
    if type(txtScript) is bs4.element.NavigableString and txtScript.find('HistoricalPriceStore') != -1:
        MBB = pd.DataFrame(columns=['date','close'])
        txtInfo = txtScript[txtScript.find('HistoricalPriceStore'):txtScript.find('}],"isPending":false,"firstTradeDate":')]
        txtInfo = "{\"" + txtInfo + "}]}}"
        objJson = json.loads(txtInfo)
        for price in objJson['HistoricalPriceStore']['prices']:
            txtDate = ""
            txtClose = ""
            for attr,val in price.items():
                if attr == "date" and val != None:
                    txtDate = datetime.datetime.fromtimestamp(val).strftime('%Y-%m-%d')
                if attr == "close" and val != None:
                    txtClose = str(val)
            if txtDate != "" and txtClose != "":
                MBB = MBB.append({"date":txtDate, "close":txtClose}, ignore_index=True)
        print(MBB)
```



```
import requests
import bs4
import json
import datetime
import pandas as pd
page = requests.get("https://finance.yahoo.com/quote/COMM.KL/history?period1=1425945600&period2=1583798400&interval=1d&filter

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


listScript = soup.find_all("script")
for script in listScript:
    txtScript = script.string
    if type(txtScript) is bs4.element.NavigableString and txtScript.find('HistoricalPriceStore') != -1:
        CIMB = pd.DataFrame(columns=['date','close'])
        txtInfo = txtScript[txtScript.find('HistoricalPriceStore'):txtScript.find('}],"isPending":false,"firstTradeDate":')]
        txtInfo = "{\"" + txtInfo + "}]}}"
        objJson = json.loads(txtInfo)
        for price in objJson['HistoricalPriceStore']['prices']:
            txtDate = ""
            txtClose = ""
            for attr,val in price.items():
                if attr == "date" and val != None:
                    txtDate = datetime.datetime.fromtimestamp(val).strftime('%Y-%m-%d')
                if attr == "close" and val != None:
                    txtClose = str(val)
            if txtDate != "" and txtClose != "":
                CIMB = CIMB.append({"date":txtDate, "close":txtClose}, ignore_index=True)
        print(CIMB)

CIMB.to_csv(r"C:/Users/L-ven Lew/Desktop/UM/Semester 4 UM/WQD 7005 Data Mining/Final Exam/Final Exam Question 1/CIMB.csv",ind
```

```
import requests
import bs4
import json
import datetime
import pandas as pd
page = requests.get("https://finance.yahoo.com/quote/RHBC.KL/history?period1=1425945600&period2=1583798400&interval=1d&filter

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


listScript = soup.find_all("script")
for script in listScript:
    txtScript = script.string
    if type(txtScript) is bs4.element.NavigableString and txtScript.find('HistoricalPriceStore') != -1:
        RHB = pd.DataFrame(columns=['date','close'])
        txtInfo = txtScript[txtScript.find('HistoricalPriceStore'):txtScript.find('}],"isPending":false,"firstTradeDate":')]
        txtInfo = "{\"" + txtInfo + "}]}}"
        objJson = json.loads(txtInfo)
        for price in objJson['HistoricalPriceStore']['prices']:
            txtDate = ""
            txtClose = ""
            for attr,val in price.items():
                if attr == "date" and val != None:
                    txtDate = datetime.datetime.fromtimestamp(val).strftime('%Y-%m-%d')
                if attr == "close" and val != None:
                    txtClose = str(val)
            if txtDate != "" and txtClose != "":
                RHB = RHB.append({"date":txtDate, "close":txtClose}, ignore_index=True)
        print(RHB)

RHB.to_csv(r"C:/Users/L-ven Lew/Desktop/UM/Semester 4 UM/WQD 7005 Data Mining/Final Exam/Final Exam Question 1/RHB.csv",index
```

```
page = requests.get("https://finance.yahoo.com/quote/%5EKLSE%3FP%3D%5EKLSE/history?period1=1425945600&period2=1583798400&inte
page.status_code

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


listScript = soup.find_all("script")
for script in listScript:
    txtScript = script.string
    if type(txtScript) is bs4.element.NavigableString and txtScript.find('HistoricalPriceStore') != -1:
        KLCI = pd.DataFrame(columns=['date','close'])
        txtInfo = txtScript[txtScript.find('HistoricalPriceStore'):txtScript.find('}],"isPending":false,"firstTradeDate":')]
        txtInfo = "{\"" + txtInfo + "}]}}"
        objJson = json.loads(txtInfo)
        for price in objJson['HistoricalPriceStore']['prices']:
            txtDate = ""
            txtClose = ""
            for attr,val in price.items():
                if attr == "date" and val != None:
                    txtDate = datetime.datetime.fromtimestamp(val).strftime('%Y-%m-%d')
                if attr == "close" and val != None:
                    txtClose = str(val)
            if txtDate != "" and txtClose != "":
                KLCI = KLCI.append({"date":txtDate, "close":txtClose}, ignore_index=True)
        print(KLCI)
KLCI.to_csv(r'C:/Users/L-ven Lew/Desktop/UM/Semester 4 UM/WQD 7005 Data Mining/Final Exam/Final Exam Question 1/KLCI3.csv',in
KLCI
```

```
import requests
import bs4
import json
import datetime
import pandas as pd
page = requests.get("https://finance.yahoo.com/quote/%5EDJI/history?period1=1425945600&period2=1583798400&interval=1d&filter=

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

listScript = soup.find_all("script")
for script in listScript:
    txtScript = script.string
    if type(txtScript) is bs4.element.NavigableString and txtScript.find('HistoricalPriceStore') != -1:
        DJI = pd.DataFrame(columns=['date','close'])
        txtInfo = txtScript[txtScript.find('HistoricalPriceStore'):txtScript.find('}],"isPending":false,"firstTradeDate":')]
        txtInfo = "{\"" + txtInfo + "}]}}"
        objJson = json.loads(txtInfo)
        for price in objJson['HistoricalPriceStore']['prices']:
            txtDate = ""
            txtClose = ""
            for attr,val in price.items():
                if attr == "date" and val != None:
                    txtDate = datetime.datetime.fromtimestamp(val).strftime('%Y-%m-%d')
                if attr == "close" and val != None:
                    txtClose = str(val)
            if txtDate != "" and txtClose != "":
                DJI = DJI.append({"date":txtDate, "close":txtClose}, ignore_index=True)
        print(DJI)

DJI.to_csv(r"C:/Users/L-ven Lew/Desktop/UM/Semester 4 UM/WQD 7005 Data Mining/Final Exam/Final Exam Question 1/DJI.csv",index
```

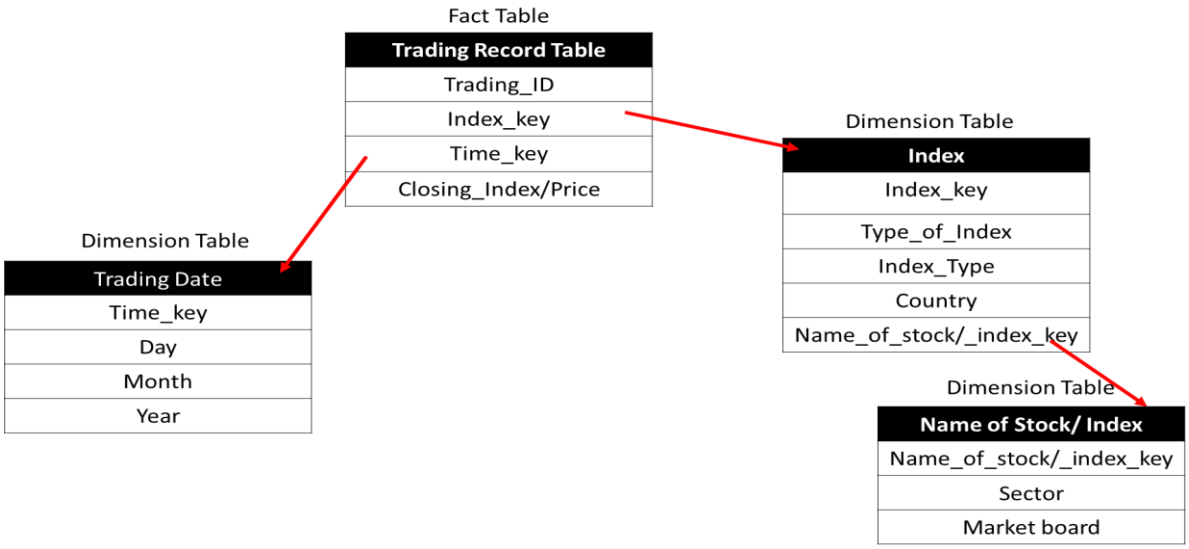
```
import requests
import bs4
import json
import datetime
import pandas as pd
page = requests.get("https://finance.yahoo.com/quote/%5EGSPC/history?period1=1425945600&period2=1583798400&interval=1d&filter
from bs4 import BeautifulSoup
soup = BeautifulSoup(page.content, 'html.parser')
#print(soup.prettify())

listScript = soup.find_all("script")
for script in listScript:
    txtScript = script.string
    if type(txtScript) is bs4.element.NavigableString and txtScript.find('HistoricalPriceStore') != -1:
        SNP = pd.DataFrame(columns=['date','close'])
        txtInfo = txtScript[txtScript.find('HistoricalPriceStore'):txtScript.find('}],"isPending":false,"firstTradeDate":')]
        txtInfo = "{\"" + txtInfo + "}]}}"
        objJson = json.loads(txtInfo)
        for price in objJson['HistoricalPriceStore']['prices']:
            txtDate = ""
            txtClose = ""
            for attr,val in price.items():
                if attr == "date" and val != None:
                    txtDate = datetime.datetime.fromtimestamp(val).strftime('%Y-%m-%d')
                if attr == "close" and val != None:
                    txtClose = str(val)
            if txtDate != "" and txtClose != "":
                SNP = SNP.append({"date":txtDate, "close":txtClose}, ignore_index=True)
        print(SNP)

SNP.to_csv(r"C:/Users/L-ven Lew/Desktop/UM/Semester 4 UM/WQD 7005 Data Mining/Final Exam/Final Exam Question 1/SNP.csv",index
SNP
```

(10 marks)

2. Draw snowflake schema diagram for the above dataset. Justify your attributes to be selected in the respective dimensions.



Index and Name of Stock/ Index dimension

Index dimension table contains Type_of_Index, Index_Type, Country and Name_of_stock/_index_key, the reason that I selected the attributes is because if I read the attributes (dataset) from this schema, only these attributes will be read, I will have a glance of what type_of_index,Index_type, Country and Name_of_stock/_index being stored in this disk. Sector and Market Board attribute are not important attribute to be read so I store it in a separate dimension namely Name of Stock/ Index dimension with the given key to retrieve the data only when it needed. In addition, it requires low disk storage and it will be faster when it is queried.

Trading Date Dimension

Trading date dimension store the day, month and year data. The reason is to ensure data storage and integrity issue when the format of the date is changed, one the date format is changed in this dimension, trading date will be consistent / standardized across all the reads.

Submissions:

Sample of SQL queries

```
# SELECT Index.Trading_ID, Type_of_Index, Country, Name of stock/ index
#    from Index_key.Index
#    join Name of stock/ index
#    on Name_of_stock/_index_key.Trading_ID = Index.Trading_ID limit 3;


DF[['Trading_Date', 'Type_of_Index', 'Country','Name of Stock/ Index','Sector','Market board']].head(3)
```

	Trading_Date	Type_of_Index	Country	Name of Stock/ Index	Sector	Market board
0	2020-03-09	Malaysia Stock	MY	MBB	Financial	Main Board
1	2020-03-06	Malaysia Stock	MY	MBB	Financial	Main Board
2	2020-03-05	Malaysia Stock	MY	MBB	Financial	Main Board

```
# SELECT index.Trading_ID, Type_of_Index, Country
#    from Index_key.index
#    join Name of Stock/ Index where Market board = "Main Board"
#    on Name_of_stock/_index_key.Trading_ID = index.Trading_ID limit 3;

DF[DF['Market board']== 'Main Board']
```

	Trading_Date	Closing_Index/Price	Type_of_Index	Sector	Market board	Month_Year	Name of Stock/ Index	Monthly_Average	Country	Index_Type
0	2020-03-09	8.24000	Malaysia Stock	Financial	Main Board	2020-03	MBB	8.423333	MY	Stock
1	2020-03-06	8.50000	Malaysia Stock	Financial	Main Board	2020-03	MBB	8.423333	MY	Stock
2	2020-03-05	8.53000	Malaysia Stock	Financial	Main Board	2020-03	MBB	8.423333	MY	Stock

The student is expected to submit answers to each question individually, and submit the document in PDF format. The student can include online materials, screenshots, videos and/or codes (ipynb format) to support your answer