Ton Duc Thang University
Faculty of Information Technology

MIDTERM PROJECT

Course: Mining Massive Datasets

Duration: 03 weeks

I. Formation

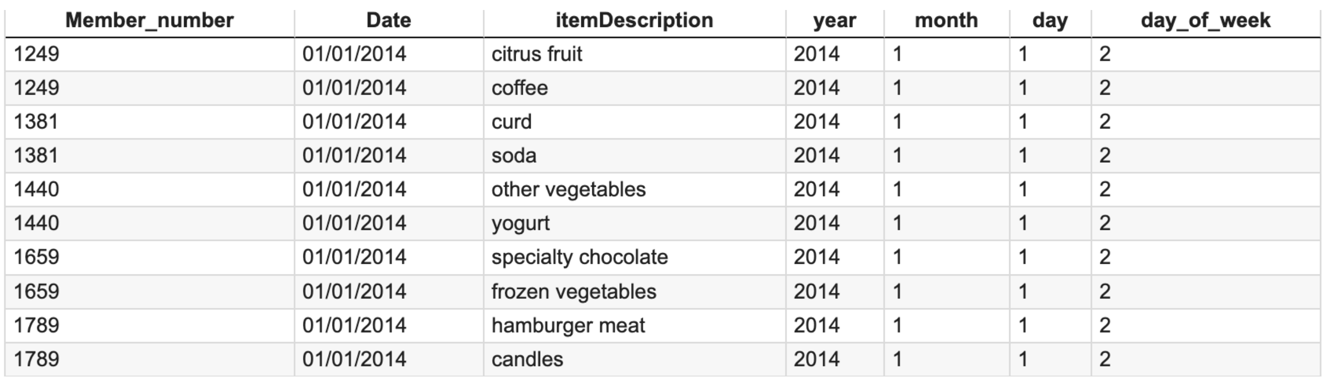
- The midterm project is conducted in groups of **04 – 05** students.
- The student group fulfills the requirements and submits the work according to the detailed instructions below.

II. Requirements

Given the **data.csv** file contains the customer's purchase data. In which, the first line contains the title (header), the remaining lines are the corresponding data rows.

- **Member_number**: member number
- **Date**: date of purchase in "dd/mm/yyyy"
- **itemDescription**: one product name
- **year**: year of purchase
- **month**: month of purchase
- **day**: day of purchase
- **day_of_week**: day of week

For example,



Member_number	Date	itemDescription	year	month	day	day_of_week
1249	01/01/2014	citrus fruit	2014	1	1	2
1249	01/01/2014	coffee	2014	1	1	2
1381	01/01/2014	curd	2014	1	1	2
1381	01/01/2014	soda	2014	1	1	2
1440	01/01/2014	other vegetables	2014	1	1	2
1440	01/01/2014	yogurt	2014	1	1	2
1659	01/01/2014	specialty chocolate	2014	1	1	2
1659	01/01/2014	frozen vegetables	2014	1	1	2
1789	01/01/2014	hamburger meat	2014	1	1	2
1789	01/01/2014	candles	2014	1	1	2

Sample data rows in data.csv (Google Colab)

a) Task 1: Item Counting

Use **textFile()** method of **SparkContext** to read **data.csv** file and count the number of items bought by each customer for each **day**, **month**, and **year**. Results are saved 03 folders named **counters_day, counters_month,** and **counters_year** respectively using **saveAsTextFile()**. Result files include:

- For day: *Member_number, Day, Quantity.*
- For month: *Member_number, Month, Quantity.*
- For year: *Member_number, Year, Quantity.*

Find out the maximum number of items in a basket sold by day, then save the results to the **max_day** folder. The result content includes *Date, Maximum number*

Notices:

- Use only **RDD** functions to find results (do not use **DataFrame**).
- The result does not contain the header line, the values on each line are separated by a "," and do not contain a "," at the end of the line.
- Students read the result file with the **DataFrame** of **SQL Context** and display it with the **show**() function to demonstrate in the exercise..

Criteria	Score
Number of items for each day	**0.5 point(s)**
Number of items for each month	**0.5 point(s)**
Number of items for each year	**0.5 point(s)**
Maximum number of items in a basket for each day	**0.5 point(s)**
Total	**2.0 point(s)**

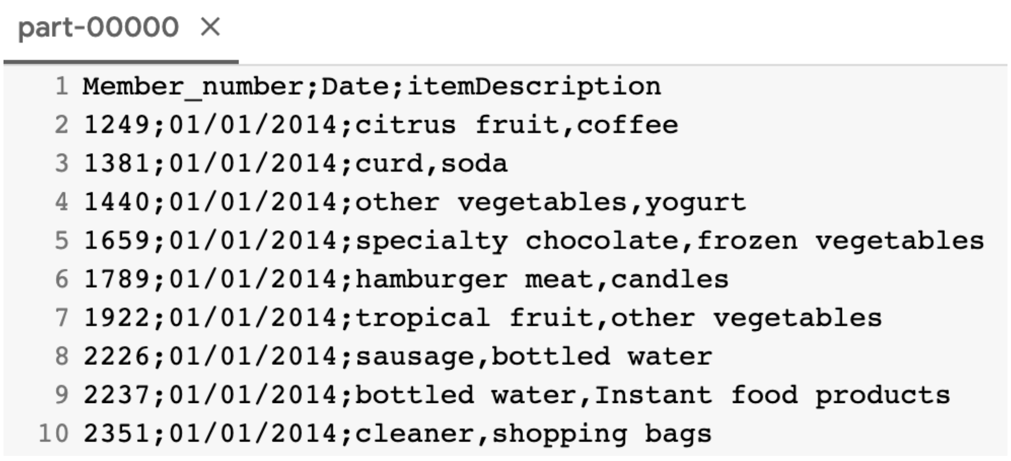
b) Task 2: Baskets

Use **textFile()** method of **SparkContext** to read **data.csv** and find out *the list of items bought by each customer for each day*. Results are saved to the **baskets** directory using the **saveAsTextFile()** method. The result content includes: *Member_number, Purchase date, List of items*.

Notices:

- Columns are separated by “;” and items are separated by “,”.
- Each item in a basket appears no more than 1 time.

For example,



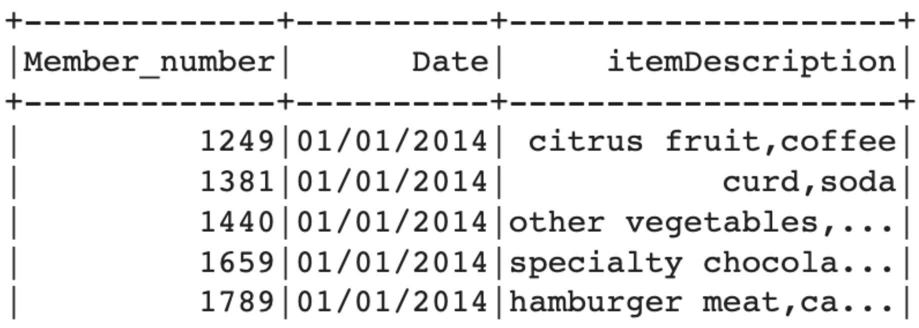
```
part-00000 ×
Member_number;Date;itemDescription
1249;01/01/2014;citrus fruit,coffee
1381;01/01/2014;curd,soda
1440;01/01/2014;other vegetables,yogurt
1659;01/01/2014;specialty chocolate,frozen vegetables
1789;01/01/2014;hamburger meat,candles
1922;01/01/2014;tropical fruit,other vegetables
2226;01/01/2014;sausage,bottled water
2237;01/01/2014;bottled water,Instant food products
2351;01/01/2014;cleaner,shopping bags
```

List of items bought by each customer for each day

The first line is the header. Items are separated by “;”

Items in a basket are separated by “,”

Students read the result file with **SQL Context DataFrame** and display it with **show**() function to demonstrate..



```
+-------------+----------+--------------------+
|Member_number|      Date|     itemDescription|
+-------------+----------+--------------------+
|         1249|01/01/2014| citrus fruit,coffee|
|         1381|01/01/2014|           curd,soda|
|         1440|01/01/2014|other vegetables,...|
|         1659|01/01/2014|specialty chocola...|
|         1789|01/01/2014|hamburger meat,ca...|
```

An example result of Task 2

Write a program (2A) that allows the user to enter a member number and a date in the form *dd/mm/yyyy*, and then print the corresponding basket.

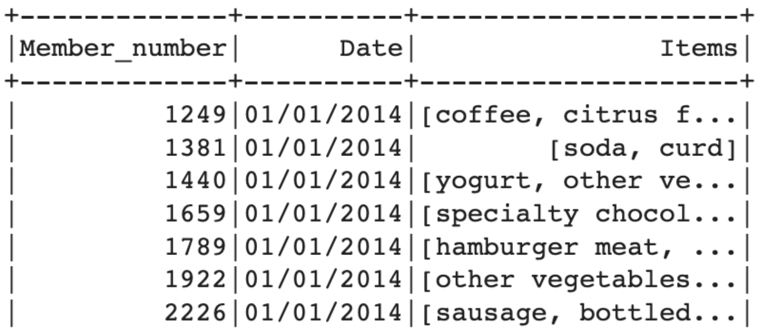
Write a program (2B) that allows the user to enter a member number and a positive integer n, and then print a result table containing the list of baskets that person has purchased in descending order of purchase date. Take only up to n lines of data.

Criteria	Score
Find out and save the list of baskets	**1.0 point(s)**
Program (2A)	**0.5 point(s)**
Program (2B)	**0.5 point(s)**
Total	**2.0 point(s)**

c) **Task 3: Frequent Itemsets**

- Read the result file in b) into a **DataFrame** named ***dfBaskets***, convert the **itemDescription** column into an array of strings and display it to the screen with the **show**() method.
- Rename columns of the result data frame as *Member_number*, *Date,* and *Items*.

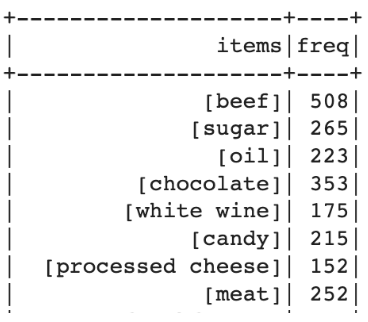
For example



```
+-------------+----------+--------------------+
|Member_number|      Date|               Items|
+-------------+----------+--------------------+
|         1249|01/01/2014|[coffee, citrus f...|
|         1381|01/01/2014|        [soda, curd]|
|         1440|01/01/2014|[yogurt, other ve...|
|         1659|01/01/2014|[specialty chocol...|
|         1789|01/01/2014|[hamburger meat, ...|
|         1922|01/01/2014|[other vegetables...|
|         2226|01/01/2014|[sausage, bottled...|
```

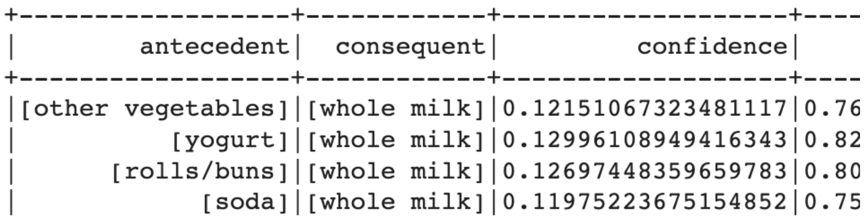
An example data frame

- Use `pyspark.ml.fpm.FPGrowth` to build up a model find frequent item sets with **minSupport=1%** and association rules with **minConfidence=10%**. Students display the list of frequent item sets and association rules to the screen using **show**() method. *For example,*



```
+--------------------+----+
|               items|freq|
+--------------------+----+
|              [beef]| 508|
|             [sugar]| 265|
|               [oil]| 223|
|         [chocolate]| 353|
|        [white wine]| 175|
|             [candy]| 215|
|  [processed cheese]| 152|
|              [meat]| 252|
```

Frequent Item sets



```
+------------------+------------+-------------------+----
|        antecedent|  consequent|         confidence|
+------------------+------------+-------------------+----
|[other vegetables]|[whole milk]|0.12151067323481117|0.76
|          [yogurt]|[whole milk]|0.12996108949416343|0.82
|      [rolls/buns]|[whole milk]|0.12697448359659783|0.80
|            [soda]|[whole milk]|0.11975223675154852|0.75
```

Association rules

Write a program (3A) that finds **support** for each item in the data set with a ***support threshold s*** entered by the user. The results are displayed on a tabular screen containing the ***item name*** column and the ***support*** column, in which the data rows are sorted descending by support.

Write a program (3B) that allows the user to enter two item names x and y. Find and print the values ***confidence(x→y)*** and ***interest(x→y)*** to the screen.

Students study and present details of `pyspark.ml.fpm.FPGrowth`.

Criteria	Score
Transform dfBaskets correctly	**0.5 point(s)**
Program (3A)	**0.5 point(s)**
Program (3B)	**0.5 point(s)**
Details of FPGrowth	**0.5 point(s)**
Total	**2.0 point(s)**

d) Task 4: Baskets-to-Vectors

- Read **data.csv** as a **DataFrame** named ***dfMembers*** with two columns
  - *Member_number*: Member number
  - *Items*: list of items, separated by ",", which contains all items bought by a customer and every item is unique.
- Students can display the results on the screen using **show**() function to demonstrate.

For example

```
+-------------+--------------------+
|Member_number|               Items|
+-------------+--------------------+
|         1249|citrus fruit,coff...|
|         1381|curd,soda,coffee,...|
|         1440|other vegetables,...|
|         1659|specialty chocola...|
```

dfMembers DataFrame

- Find the item list in the entire data set, save it in the ***items*** variable, where the elements are unique and sorted in ascending alphabetical order. Then, create a dictionary named

dictItems containing pairs *<item name>:<list index>*. Print the values of the two variables above to the screen.

For example

```
['Instant food products', 'UHT-milk', 'abrasive cleaner', 'artif.
{'Instant food products': 0, 'UHT-milk': 1, 'abrasive cleaner': 2,
```

*Examples of **items** and **dictItems***

- Implement the function ***basket2vector(member, basket, dictItems)*** taking a member number (***member***), a list of items (***basket***) separated by ",", and ***dictItems*** above. The function returns a **Vectors.sparse** (**pyspark.ml.linalg**), in form of multi-hot coding, with the same length as ***dictItems***, elements are 0.0 or 1.0 corresponding to existing items in ***basket*** *(for example there is item **it** in **basket**, then **dictItems[it]** = 1.0).*

*For example: invoke **basket2vector** and pass the first row of **dfMembers** in*

```
print(basket2vector(dfMembers.first()['Member_number'],
                    dfMembers.first()['Items'],
                    dictItems))
```

```
(167,[11,30,34,61,138],[1.0,1.0,1.0,1.0,1.0])
```

*Returned value from **basket2vector** for the first row of **dfMembers***

Students note that **DataFrame *dfMembers*** is also used in e).

Criteria	Score
Find out the item list *Create the dictionary correctly*	**0.5 point(s)**
Function basket2vector *Create dfMember correctly*	**0.5 point(s)**
Total	**1.0 point(s)**

e) Task 5: Similar baskets

- Create a model **MinHashLSH** in module **pyspark.ml.feature** to find out customers with similar shopping habit. Use the data frame ***dfMembers*** from Task 4, in which
  - ***inputCol*** is "Items"

- ***outputCol*** is "Hashes"
- Number of hash tables is 08

- Train the model with ***dfMembers***, apply *transform*() function in ***dfMembers***, and then display the result data frame on the screen.
- Note do not change the value of ***dfMembers*** because it is used in later tasks**.**

For example

```
+-------------+--------------------+--------------------+
|Member_number|               Items|              Hashes|
+-------------+--------------------+--------------------+
|         1249|(167,[11,30,34,61...|[[2.85001106E8], ...|
|         1381|(167,[1,10,11,28,...|[[3.9022841E7], [...|
|         1440|(167,[28,64,102,1...|[[3.41446049E8], ...|
|         1659|(167,[12,14,26,34...|[[1.02200615E8], ...|
```

Result ***DataFrame*** *transformed from* ***dfMembers***

- Use **approxSimilarityJoin**() to find out pairs of customers with similar shopping habit, where **JaccardDistance** is not over ***0.5***. Display the result data frame with columns including member numbers and **JaccardDistance**, in which selecting rows with positive *JaccardDistance*.

For example,

```
+----+----+-------------------+
| idA| idB|    JaccardDistance|
+----+----+-------------------+
|3124|1063|               0.25|
|1643|4535|0.19999999999999996|
|1860|3605|               0.25|
|4342|1056| 0.2857142857142857|
|2911|3714|               0.25|
|3715|4805|               0.25|
```

Result pairs of customers

- Use **approxNearestNeighbors**() to find out **05** customers whose shopping habit is the most similar to the one whose *member_number* is input by users. Display the result on the screen.

For example,

```
+-------------+--------------------+--------------------+------------------+
|Member_number|               Items|              Hashes|           distCol|
+-------------+--------------------+--------------------+------------------+
|         1249|(167,[11,30,34,61...|[[2.85001106E8], ...|               0.0|
|         1321|(167,[11,30,138],...|[[2.85001106E8], ...|               0.4|
|         1263|(167,[11,30,61,10...|[[2.85001106E8], ...|               0.5|
|         1794|(167,[11,30,138,1...|[[2.85001106E8], ...|0.5714285714285714|
|         4327|(167,[30,34,63,76...|[[3.43690326E8], ...|0.5714285714285714|
+-------------+--------------------+--------------------+------------------+
```

Top 05 customers whose shopping habit is the most similar to the first one in ***dfMembers***

Students study and present details of **MinHashLSH** in PySpark.

Criteria	Score
Use MinHashLSH to transform data correctly	**0.5 point(s)**
Find out pairs of customers with similar shopping habit	**0.5 point(s)**
Find out top 05 most similar customers to input one	**0.5 point(s)**
Study and present details of MinHashLSH	**0.5 point(s)**
Total	**2.0 point(s)**

f) Task 7 (1.0 point): Presentation

- Student groups compose a presentation to report your work.
- **THERE IS NO PRESENTATION TEMPLATES. STUDENTS ARANGE CONTENTS IN A LOGICAL LAYOUT BY YOURSELVES.**
- The presentation must include below contents
  - Student list: Student ID, Full name, Email, Assigned tasks, Complete percentage.
  - Briefly present approaches to solve tasks, should make use of pseudo code/diagrams.
  - AVOID EMBEDDING RAW SOURCE CODE IN THE PRESENTATION.
  - Study topics are introduced briefly with practical examples.
  - Advantages versus disadvantages
  - A table of complete percentages for each task.
  - References are presented in IEEE format.

- **Format requirements:** slide ratio of 4x3, avoid using dark background/colorful shapes because of projector quality, students ensure contents are clear enough when printing the presentation in grayscale.
- Presentation duration is **10 minutes**.

III. Submission Instructions

- Create a folder whose name is as

  <Student ID 1>_< Student ID 2>_< Student ID 3>_< Student ID 4>
- Content:
  - **source.ipynb** → source code (remain all cell outputs)
  - **source.pdf** → pdf of the notebook
  - **presentation.pdf** → presentation.
- Compress the folder to a zip file and submit by the deadline.

IV. Policy

- **Student groups submitting late get 0.0 points for each member.**
- **Wrong student IDs in the submission filename cause 0.0 points for the corresponding students.**
- **Missing required materials in the submission loses at least 50% points of the presentation.**
- **Copying source code on the internet/other students, sharing your work with other groups, etc. cause 0.0 points for all related groups.**
- **If there exist any signs of illegal copying or sharing of the assignment, then extra interviews are conducted to verify student groups' work.**

-- THE END --