Vietnam General Confederation of Labor
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

MIDTERM EASSY

MINING MASSIVE DATA SETS

Instructor: **Mr. NGUYEN THANH AN**

Student: **Do Minh Quan – 521H0290**

Ho Huu An – 521H0489

Van Cong Nguyen Phong – 521H0287

Nguyen Le Phuoc Tien-521H0514

Class **: 21H50301**

Year **: 25**

HO CHI MINH CITY, 2023

Vietnam General Confederation of Labor
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

MIDTERM EASSY

MINING MASSIVE DATA SETS

Instructor: **Mr. NGUYEN THANH AN**

Student: **Do Minh Quan – 521H0290**

Ho Huu An – 521H0489

Van Cong Nguyen Phong – 521H0287

Nguyen Le Phuoc Tien-521H0514

Class **: 21H50301**

Year **: 25**

HO CHI MINH CITY, 2023

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mr. Nguyen Thanh An from Ton Duc Thang University for his invaluable support and guidance throughout the development of this web application. His expertise in the field of Mining Massive Data has been instrumental in shaping the direction of this project.

Mr. Nguyen Thanh An's commitment to excellence and his willingness to share his knowledge have significantly contributed to the success of this endeavor. His mentorship has not only enhanced my understanding of Node.js and web application development but has also inspired me to explore new horizons in the realm of real-time communication over the web.

I extend my heartfelt thanks to Mr. NGUYEN THANH AN for his continuous encouragement and support, which have played a pivotal role in the completion of this project. His dedication to fostering a learning environment has been a source of inspiration, and I am grateful for the opportunity to work under his guidance.

Ho Chi Minh city, 4th December, 2023

Author

(Sign and write full name)

Do Minh Quan

THIS PROJECT WAS COMPLETED AT
TON DUC THANG UNIVERSIY

I fully declare that this is my own project and is guided by Mr. NGUYEN THANH AN; The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

Besides that, the project also uses a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

Should any frauds were found, I will take full responsibility for the content of my report. Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

Ho Chi Minh city, 6th March, 2024

Author

(Sign and write full name)

Do Minh Quan
(Trưởng nhóm)

CONFIRMATION AND ASSESSMENT SECTION

Instructor confirmation section

_____

_____

_____

_____

_____

_____

_____

Ho Chi Minh January, 2024

(Sign and write full name)

Evaluation section for grading instructor

_____

_____

_____

_____

_____

_____

_____

Ho Chi Minh January 2024

(Sign and write full name)

SUMMARY

This essay delves into the realm of mining massive data by leveraging PySpark's capabilities, specifically focusing on RDDs, DataFrames, and the PCY algorithm. The objective is to extract meaningful insights efficiently from vast datasets while maintaining scalability and performance.

RDDs, offering fault tolerance and parallel processing, form the backbone of the analysis pipeline. They provide a resilient foundation for processing large volumes of data in distributed environments. DataFrames, on the other hand, offer a higher-level abstraction, enabling easier manipulation and optimization of data processing tasks.

Incorporating the PCY algorithm, renowned for its efficiency in mining frequent itemsets, further enhances the analysis process. By integrating this algorithm with PySpark's RDDs and DataFrames, the essay aims to streamline the extraction of valuable patterns and associations from massive datasets.

The essay explores practical implementation strategies, emphasizing preprocessing, transformation, and analysis of data using RDDs and DataFrames. Additionally, it delves into the intricacies of integrating the PCY algorithm within the PySpark framework, highlighting its role in enhancing the efficiency of frequent itemset mining.

Through empirical evaluation and case studies, the effectiveness of the proposed approach is demonstrated, showcasing the potential of RDDs, DataFrames, and the PCY algorithm in mining massive data. The essay concludes by underscoring the significance of this integrated approach in handling large-scale data mining tasks, paving the way for more efficient and scalable data analysis pipelines.

INDEX

Contents

LIST OF ABBREVIATIONS

1. RDD : Resilient Distributed Dataset
2. PCY : Park-Chen-Yu Algorithm

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 – INTRODUCTION

1.1 PySPark

PySpark is an interface for Apache Spark in Python. With PySpark, you can write Python and SQL-like commands to manipulate and analyze data in a distributed processing environment.

Figure 1: PySpark

Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark Community released a tool, PySpark. Using PySpark, you can work with RDDs in Python programming language also. It is because of a library called Py4j that they are able to achieve this.

PySpark offers PySpark Shell which links the Python API to the spark core and initializes the Spark context. Majority of data scientists and analytics experts today use Python because of its rich library set. Integrating Python with Spark is a boon to them.

1.2 Resilient Distributed Datasets

1.2.1 Introduction

Resilient Distributed Datasets (RDDs) are the fundamental building blocks of Pyspark which are a distributed memory abstraction that helps a programmer to perform in-memory computations on large clusters that too in a fault-tolerant manner. In this tutorial we will be focussing on Introduction, Features of RDD, Pair RDDs, Transformations and actions of RDD and other concepts

RDDs are a collection of objects similar to a list in Python, with the difference being RDD is computed on several processes scattered across multiple physical servers also called nodes in a cluster while a Python collection lives and processes in just one process. It provides parallelism by default.

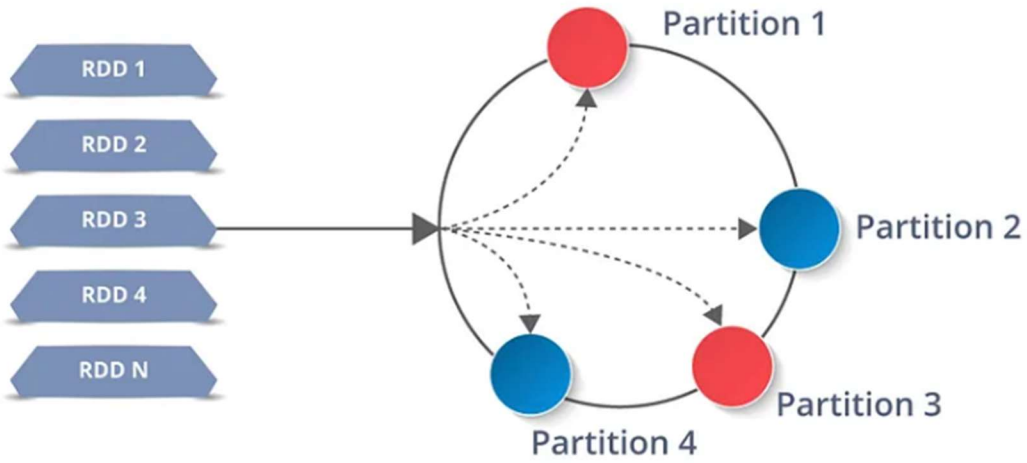


Figure 2: RDD

1.2.2 Features of RDD

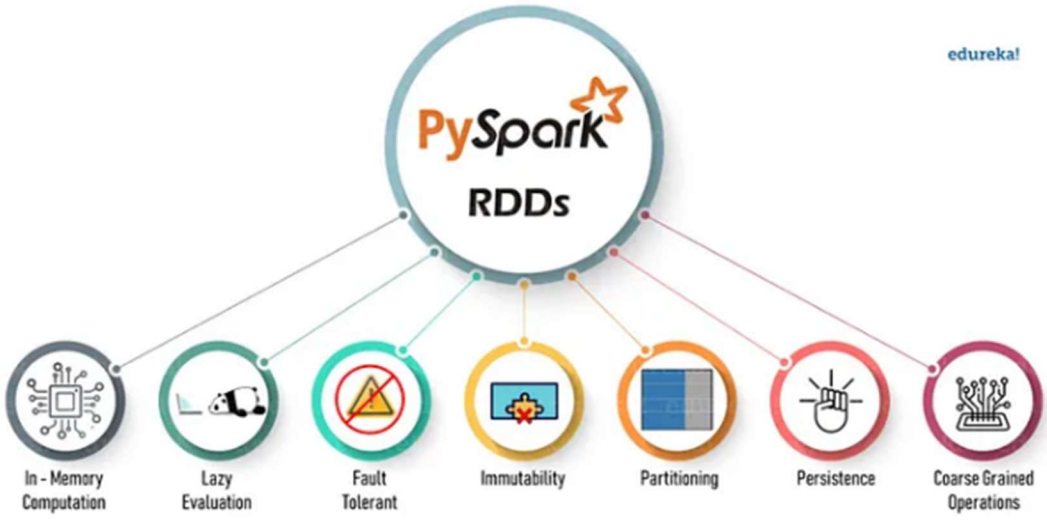


Figure 3: Feature of RDD

- In-Memory Computation: RDDs allow computations to be performed in-memory, which means that intermediate results are stored in memory rather than being written to disk after each transformation. This enables faster processing by reducing the overhead of disk I/O.
- Lazy Evaluation: RDDs employ lazy evaluation, meaning that transformations are not immediately executed. Instead, transformations are stored as a lineage graph, and actions trigger the execution of the entire lineage graph. This optimization improves performance by allowing Spark to optimize the execution plan and minimize unnecessary computations.

- Fault Tolerance: RDDs are fault-tolerant, meaning that they can recover from failures automatically. This is achieved through lineage information stored within RDDs, which allows Spark to recompute lost partitions in case of failures.
- Immutability: RDDs are immutable, meaning that once created, their content cannot be changed. Instead, transformations create new RDDs, preserving the original data. Immutability simplifies fault tolerance and enables optimizations like lineage tracking.

- Partitioning: RDDs partition data across the cluster to enable parallel processing. Each partition of an RDD is processed independently on different nodes, allowing for efficient distributed computation. Partitioning can be controlled manually or automatically by Spark.
- Persistence: RDDs can be persisted in memory or disk to avoid recomputation. Spark provides different levels of persistence, allowing developers to choose the appropriate level based on the characteristics of their workload and available resources.

- Coarse-Grained Operations: RDDs support coarse-grained operations, where entire datasets are processed at once. This contrasts with fine-grained operations, which operate on individual elements. Coarse-grained operations are typically more efficient in distributed systems, as they reduce communication overhead and enable better parallelism.

1.2.3 Practical examples

Suppose you have a dataset containing information about sales transactions. You can create an RDD from this dataset and perform transformations and actions on it. For example, you can filter out transactions over a certain amount, map the data to extract specific fields, and then perform aggregation operations like summing up the total sales amount.

1.3 DataFrame

1.3.1 Introduction

This is a short introduction and quickstart for the PySpark DataFrame API. PySpark DataFrames are lazily evaluated. They are implemented on top of RDDs. When Spark transforms data, it does not immediately compute the transformation but plans how to compute later. When actions such as collect() are explicitly called, the computation starts. This notebook shows the basic usages of the DataFrame, geared mainly for new users. You can run the latest version of these examples by yourself in 'Live Notebook: DataFrame' at the quickstart page.

There is also other useful information in Apache Spark documentation site, see the latest version of Spark SQL and DataFrames, RDD Programming Guide, Structured Streaming Programming Guide, Spark Streaming Programming Guide and Machine Learning Library (MLlib) Guide.

PySpark applications start with initializing SparkSession which is the entry point of PySpark as below. In case of running it in PySpark shell via pyspark executable, the shell automatically creates the session in the variable spark for users.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

1.3.2 Practical examples

Consider a scenario where you have a dataset containing information about e-commerce transactions, including details about customers, products, and purchase amounts. You can create a DataFrame from this dataset and then perform operations like filtering to find transactions made by a specific customer, joining with another DataFrame to enrich the data with additional information, and aggregating to calculate total sales revenue.

1.4 Park-Chen-Yu algorithm (PCY)

1.4.1 Introduction

The PCY algorithm (Park-Chen-Yu algorithm) is a data mining algorithm that is used to find frequent itemets in large datasets. It is an improvement over the Apriori algorithm and was first described in 2001 in a paper titled “PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth” by Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto.

The PCY algorithm uses hashing to efficiently count item set frequencies and reduce overall computational cost. The basic idea is to use a hash function to map itemsets to hash buckets, followed by a hash table to count the frequency of itemsets in each bucket.

1.4.2 Practical examples

Suppose you have a dataset containing records of customer transactions in a supermarket, where each transaction consists of a set of items purchased by a customer.

By applying the PCY algorithm, you can efficiently identify frequent itemsets, such as commonly purchased combinations of products. This information can be valuable for various purposes, such as market basket analysis and personalized recommendations.

CHAPTER 2 – IMPLEMENT

2.1 Diagrams

2.1.1 Task 1: RDD

Input: Path to baskets.csv
Ouput : Print results on the screen and save them to folder **f1**
Function f1:

- ✓ Find the list of distinct products.
- ✓ Results are sorted in the ascending order of product names.
- ✓ Print down 10 frist and 10 last products in the resulting list.

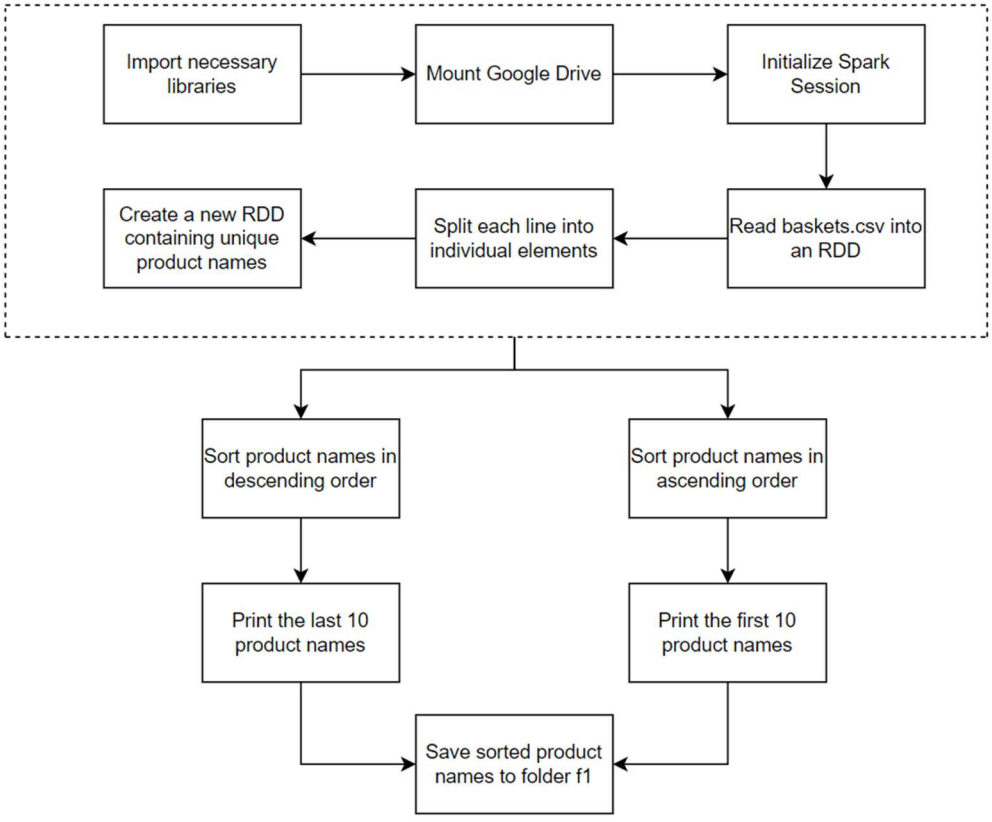


Figure 4: Diagram f1

Input: Path to baskets.csv
Ouput : Print results on the screen and save them to folder **f2**
Function f2:

- ✓ Find the list of distinct products and their frequency of being purchased.
- ✓ Results are sorted in the descending order of frequency.
- ✓ Select top 100 products with the highest frequency, draw a bar chart to visualize their frequency.

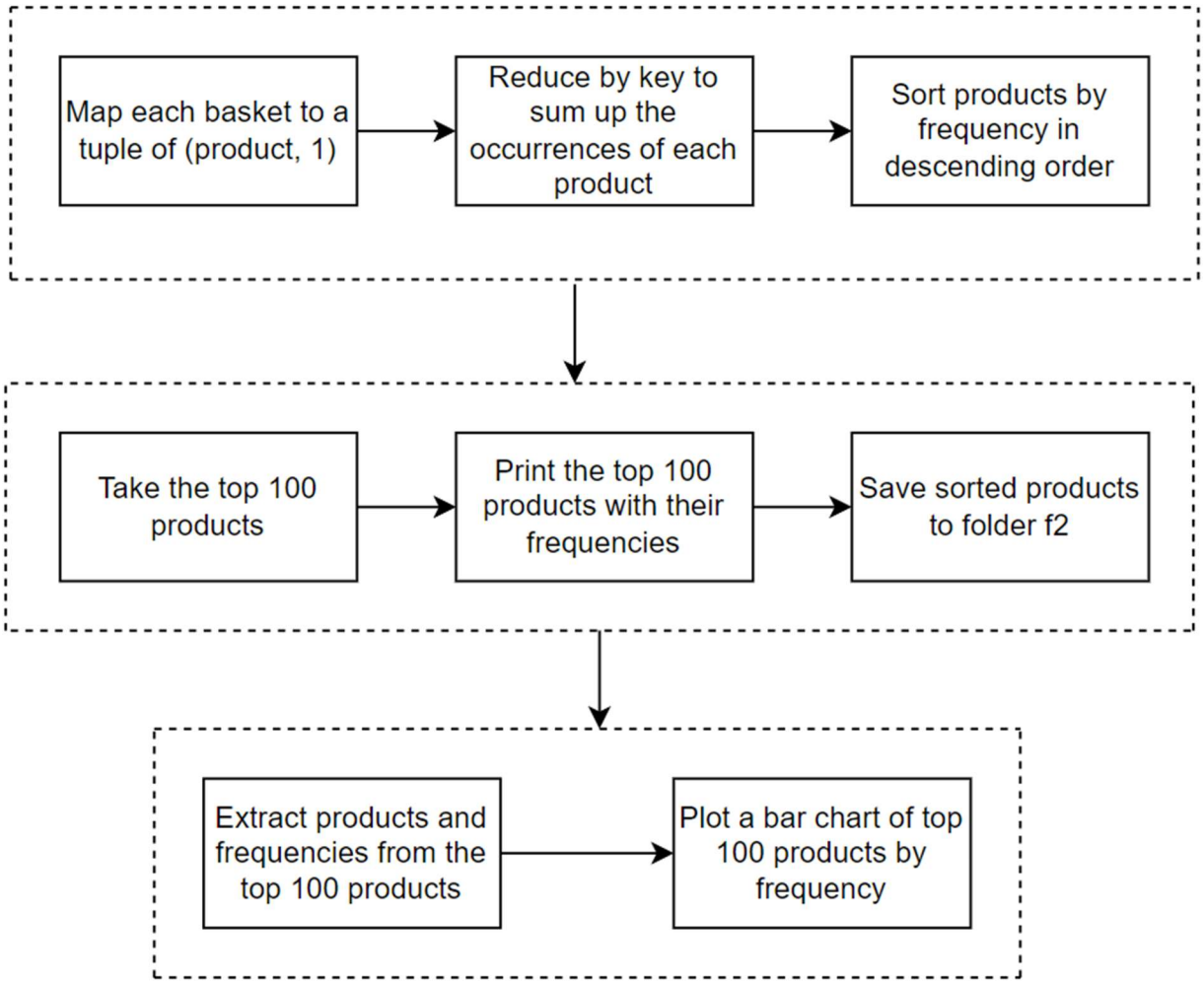


Figure 5: Diagram f2

Input: Path to baskets.csv
Ouput : Print results on the screen and save them to folder **f3**
Function f3:

- ✓ Find the number of baskets for each member.
- ✓ A basket is a set of distinct products bought by a member in a date.
- ✓ Results are sorted in the descending order of number of baskets.
- ✓ Select top 100 members with the largest number of baskets, draw a bar chart to visualize their number of baskets

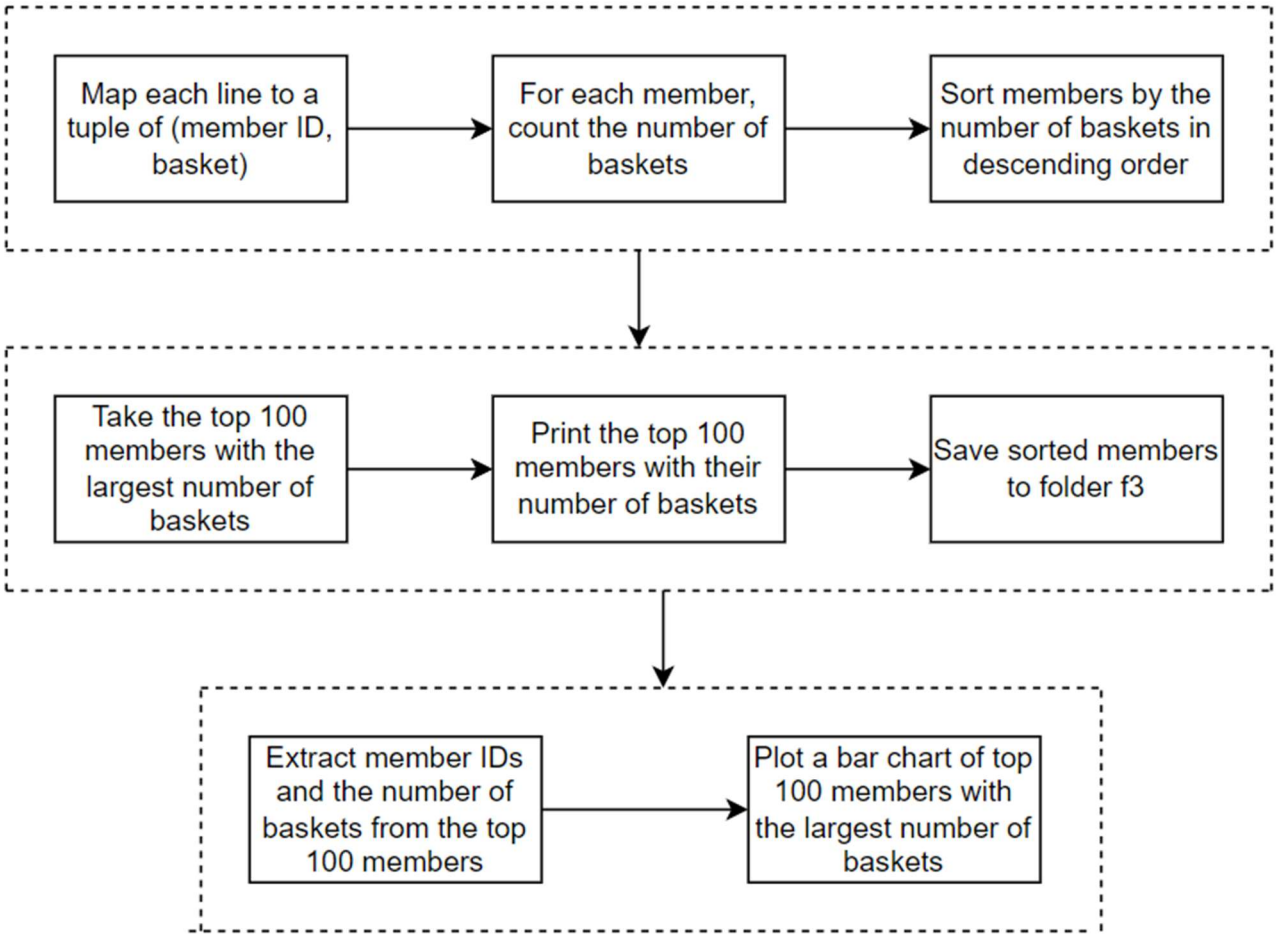


Figure 6: Diagram f3

Input: Path to baskets.csv
Ouput : Print results on the screen and save them to folder **f4**
Function f4:

- ✓ Find the member that bought the largest number of distinct products.
- ✓ Print down the member number and the number of products.
- ✓ Find the product that is bought by the most members.
- ✓ Print down its name and the number of members.

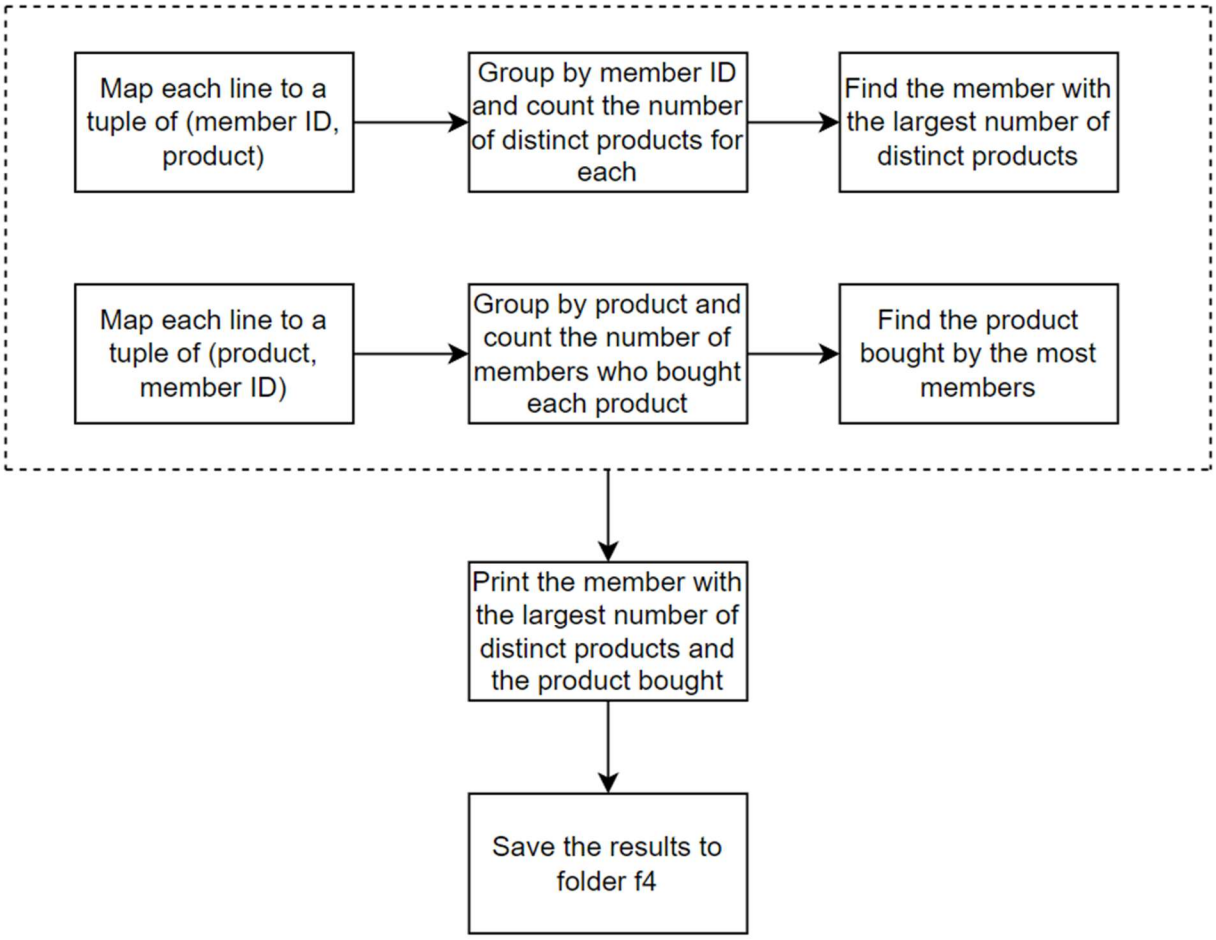


Figure 7: Diagram f4

2.1.2 Task 2: DATAFRAME

Use DataFrame (PySpark) to find out the list of baskets. A basket is a set of products bought by a member in a date. Resulting baskets are sorted in the ascending order of year, month, day.

With the resulting DataFrame, find the number of baskets bought in each date. Draw a line chart to visualize the result.

Save the resulting baskets in the folder baskets.

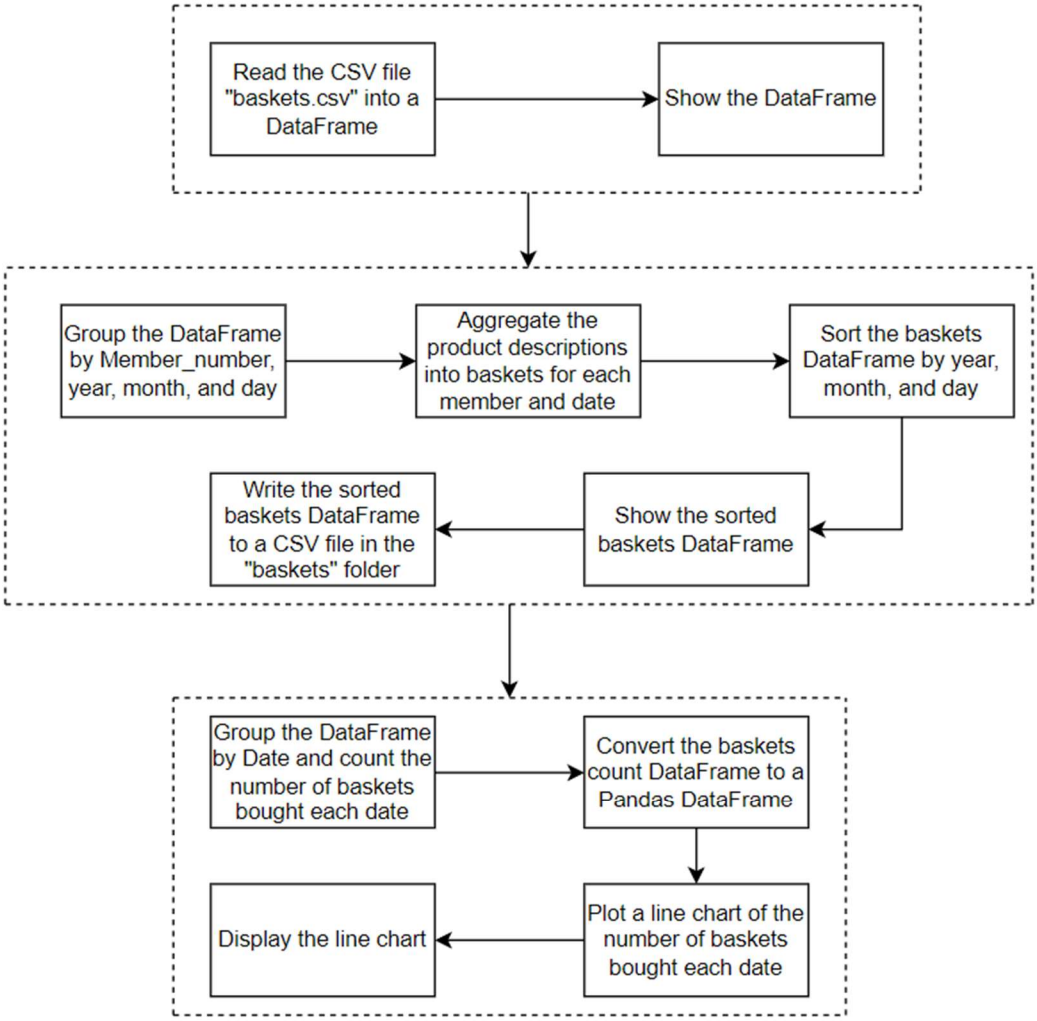


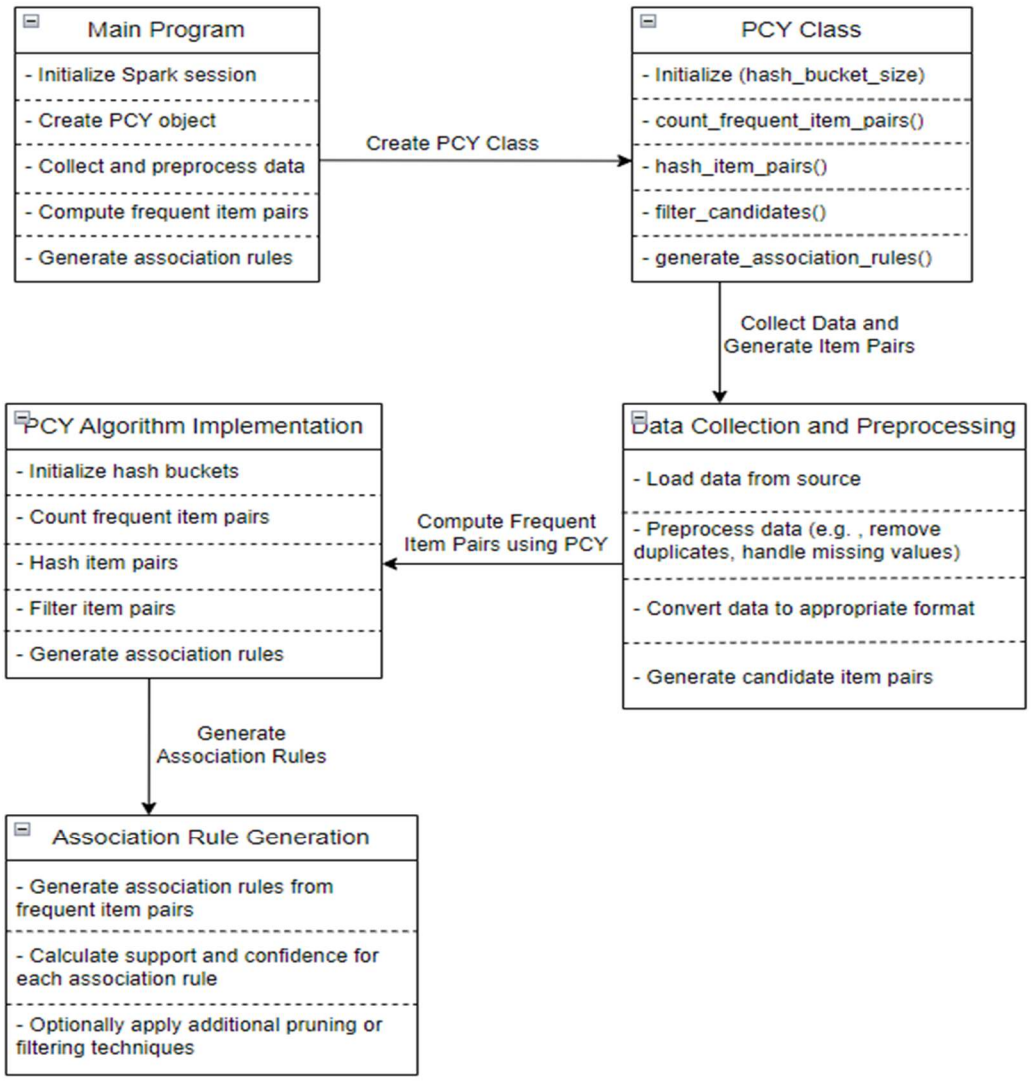
Figure 8: Diagram task 2

2.1.3 Task 3: PCY

Constructor: receives a path to a file consisting of baskets from task 2; constant **s** is the support threshold (i.e., s = 0.3); constant c is the confidence threshold (i.e., c = 0.5).

run(): run the algorithm. After that,

- ✓ Save the resulted DataFrame consisting of frequent pairs to **pcy_frequent_pairs.csv**
- ✓ Save the resulted DataFrame consisting of association rules to **pcy_association_rules.csv**.
- ✓ Schemas of DataFrames are based on the one of **FPGrowth**.



CHAPTER 3 – EVALUATION

3.1 Task assigments

Full name	Assigned tasks (100%)	Completion level
Do Minh Quan	25%	On schedule as assigned
Ho Huu An	25%	On schedule as assigned
Van Cong Nguyen Phong	25%	On schedule as assigned
Nguyen Le Phuoc Tien	25%	On schedule as assigned

Table 1: Task assignments

3.2 Self- assessment

TASK 1 : RDD	Incomplete	Complete	Complete Percentage
Function f1	FALSE	TRUE	100%
Function f2	FALSE	TRUE	100%
Function f3	FALSE	TRUE	100%
Function f4	FALSE	TRUE	100%
TASK 2: DATAFRAME			
Resulting baskets are sorted in the ascending order of year, month, day	FALSE	TRUE	100%
Draw a line chart to visualize the result.	FALSE	TRUE	100%
Save the resulting baskets in the folder baskets.	FALSE	TRUE	100%
TASK 3: PCY			

Save the resulted DataFrame consisting of frequent pairs to pcy_frequent_pairs.csv	FALSE	TRUE	100%
Save the resulted DataFrame consisting of association rules to pcy_association_rules.csv	FALSE	TRUE	100%
Schemas of DataFrames are based on the one of FPGrowth	FALSE	TRUE	100%

Table 2: Self-Assessment

CHAPTER 4 – REFERENCES

1) Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93), (pp. 207-216).

2) Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), (pp. 487-499).

3) Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00), (pp. 1-12).

4) Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann.

5) Tan, P. N., Steinbach, M., & Kumar, V. (2006). Introduction to Data Mining. Pearson.

6) Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

7) Zaki, M. J., & Meira, W. Jr. (2014). Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press.