

**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, 2024

**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, 2024

ACKNOWLEDGEMENT

We 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.

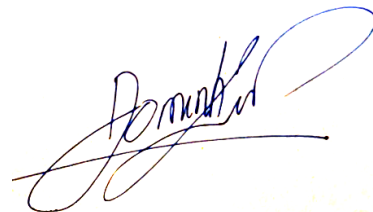
Mr. Nguyen Thanh An's commitment to excellence and his willingness to share his knowledge have significantly contributed to the success of this endeavor. His mentorship has not only enhanced my understanding of Mining Massive Data but has also inspired me to explore new horizons in the realm of data processing, learning more ways to solve problems.

We would like to send our sincere 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 we are grateful for the opportunity to work under his guidance.

Ho Chi Minh city, 14th March, 2024

Author

(Sign and write full name)

A handwritten signature in blue ink, appearing to read 'Do Minh Quan', with a stylized, flowing script.

Do Minh Quan

THIS PROJECT WAS COMPLETED AT TON DUC THANG UNIVERSITY

We fully declare that this is our 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.

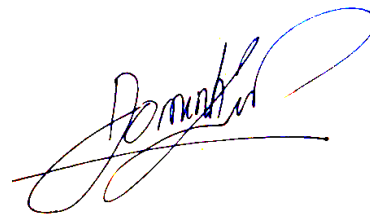
Besides that, the project also uses a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

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, 14th March, 2024

Author

(Sign and write full name)



Do Minh Quan

CONFIRMATION AND ASSESSMENT SECTION

Instructor confirmation section

Ho Chi Minh March, 2024
(Sign and write full name)

Evaluation section for grading instructor

Ho Chi Minh March, 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

ACKNOWLEDGEMENT	i
CONFIRMATION AND ASSESSMENT SECTION	iii
SUMMARY	iv
LIST OF ABBREVIATIONS	3
LIST OF FIGURES	4
LIST OF TABLES	5
CHAPTER 1 – INTRODUCTION	6
1.1 PySpark	6
1.2 Resilient Distributed Datasets	6
1.2.1 Introduction	6
1.2.2 Features of RDD	7
1.2.3 Practical examples	9
1.3 DataFrame	9
1.3.1 Introduction	9
1.3.2 Practical examples	10
1.4 Park-Chen-Yu algorithm (PCY)	10
1.4.1 Introduction	10
1.4.2 Practical examples	11
CHAPTER 2 – IMPLEMENTATION	12
2.1 Task 1: RDD	12
2.2 Task 2: DATAFRAME	16
2.3 Task 3: PCY	16
CHAPTER 3 – EVALUATION	19
3.1 Advantages versus disadvantages	19

3.2 Task assignments	20
3.3 Self- assessment	20
CHAPTER 4 – REFERENCES	22

LIST OF ABBREVIATIONS

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

LIST OF FIGURES

Figure 1: PySpark.....	6
Figure 2: RDD.....	7
Figure 3: Feature of RDD	7
Figure 4: Diagram f1	12
Figure 5: Diagram f2.....	13
Figure 6: Diagram f3.....	14
Figure 7: Diagram f4.....	15

LIST OF TABLES

Table 1: Task assignments	20
Table 2: Self-Assessment	21

CHAPTER 1 – INTRODUCTION

1.1 PySpark

PySpark is an interface for Apache Spark in Python. With PySpark, Python and SQL-like commands can be written 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, working 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.

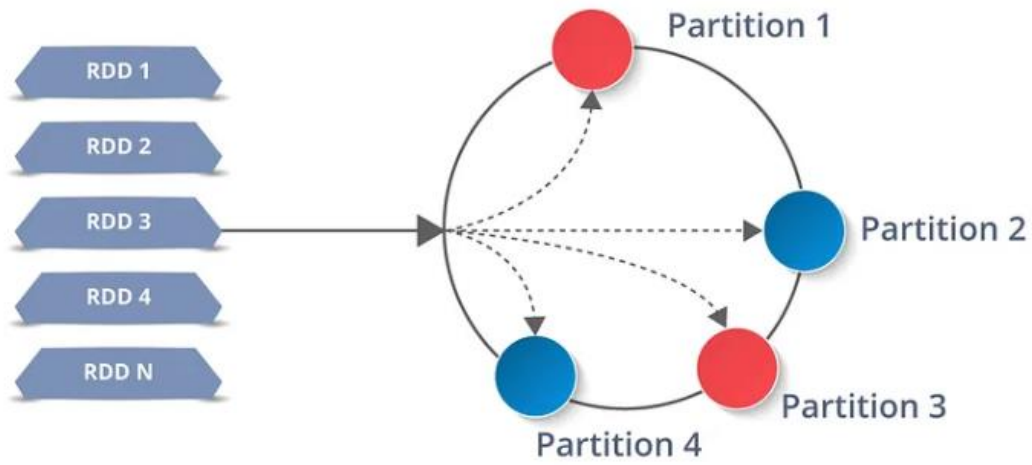


Figure 2: RDD

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 project has a dataset containing information about sales transactions. The general public is likely to create an RDD from this dataset and perform transformations and actions on it. For example, workers have the ability to 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

A DataFrame in PySpark is a distributed collection of data organized into named columns. It is a fundamental data structure that allows for efficient processing of large datasets across multiple machines. PySpark DataFrames can be created from various data sources such as external databases, structured data files, or existing resilient distributed datasets (RDDs) [1].

Key Features of PySpark DataFrames:

- 1) **Tabular Structure:** PySpark DataFrames are organized in a tabular structure with rows and columns, similar to traditional database tables.
- 2) **Column Names and Types:** Each column in a DataFrame has a name and a specific data type associated with it.

- 3) Lazy Evaluation: PySpark DataFrames use lazy evaluation, which means that operations on the DataFrame are not executed immediately. Instead, they are evaluated only when an action is triggered.
- 4) Immutable: PySpark DataFrames are immutable, meaning that they cannot be modified once created. This immutability ensures data consistency and enables efficient processing.
- 5) Language Support: PySpark DataFrames have API support for multiple programming languages, including Python, R, Scala, and Java.
- 6) Scalability: PySpark DataFrames can handle large-scale datasets, including petabytes of data, by leveraging the distributed computing capabilities of Apache Spark.
- 7) Performance Optimization: PySpark DataFrames provide optimized memory management and execution plans, resulting in improved performance compared to traditional RDDs.

1.3.2 Practical examples

There is a dataset available that contains various information related to e-commerce transactions. This includes details about customers, products, and the amounts spent on purchases. By utilizing this dataset, it is possible to create a DataFrame, which is a structured data format commonly used in programming and data analysis. With the DataFrame, it becomes feasible to carry out operations such as filtering the data to identify transactions made by a particular customer, merging the DataFrame with another one to enhance the available information, and performing aggregations to determine the total revenue generated from sales.

1.4 Park-Chen-Yu algorithm (PCY)

1.4.1 Introduction

The Park-Chen-Yu algorithm is a numerical method used for solving nonlinear systems of equations. It was developed by Park, Chen, and Yu in 1988 as an extension

of the Newton-Raphson method, specifically designed to handle highly nonlinear systems.

The algorithm is particularly useful when the traditional Newton-Raphson method encounters convergence issues or fails to converge altogether. It employs a modified Jacobian matrix and incorporates damping factors to improve stability and convergence properties.

The PCY algorithm iteratively updates the solution vector by solving a linearized system of equations, similar to the Newton-Raphson method. However, it introduces additional modifications to the solution update formula, which help to prevent divergence and improve convergence behavior.

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

The PCY algorithm applied to e-commerce data helps identify frequent event sequences in customer transactions. This information can be used to optimize website layout, recommend related products, improve inventory management, personalize marketing campaigns, and enhance customer retention strategies.

The PCY algorithm also efficiently identifies frequent event sequences in network traffic by using a memory-efficient counting technique. It helps understand network behavior, optimize performance, and enhance security measures

CHAPTER 2 – IMPLEMENTATION

2.1 Task 1: RDD

Function f1:

Input: Path to baskets.csv

Output : Print results on the screen and save them to folder **f1**

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

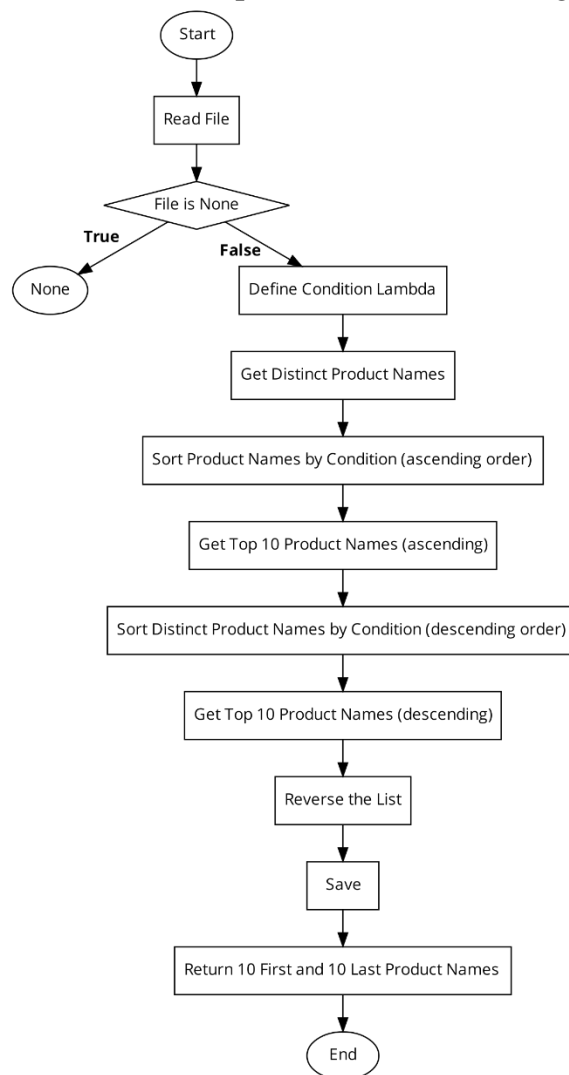


Figure 4: Diagram f1

Function f2:

Input: Path to baskets.csv

Output : Print results on the screen and save them to folder **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.

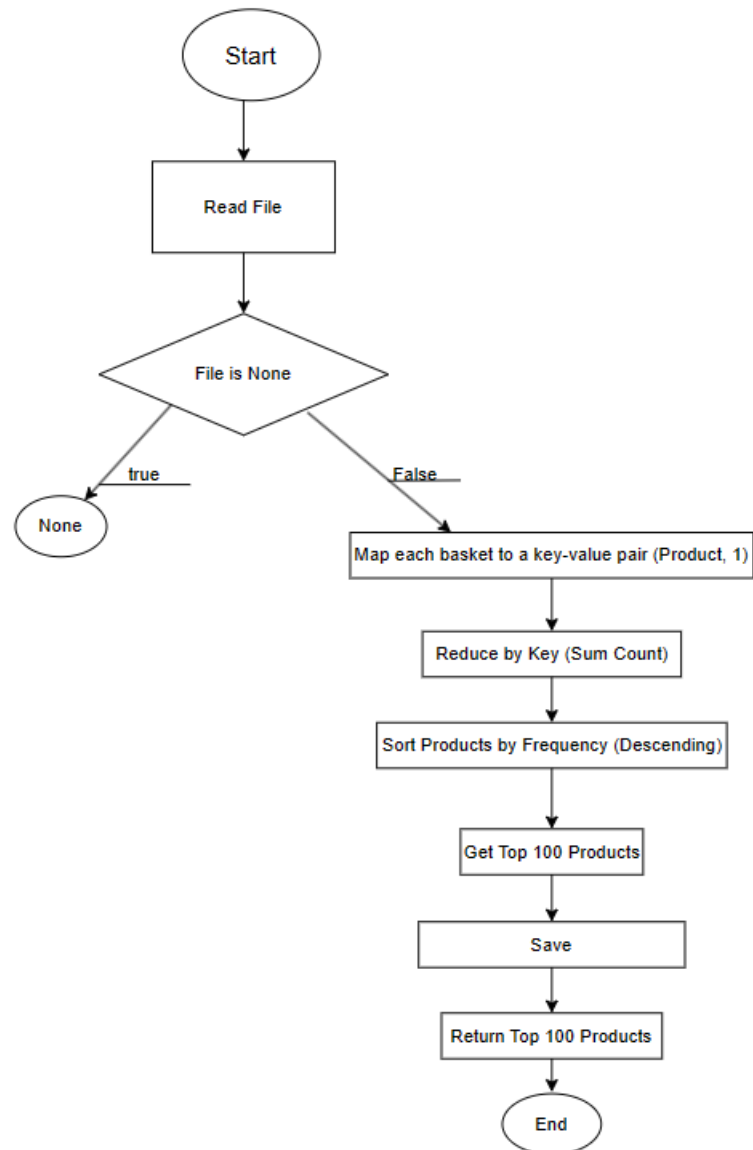


Figure 5: Diagram f2

Function F3

Input: Path to baskets.csv

Output : Print results on the screen and save them to folder **f3**

- ✓ Find the number of baskets for each member.
- ✓ 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

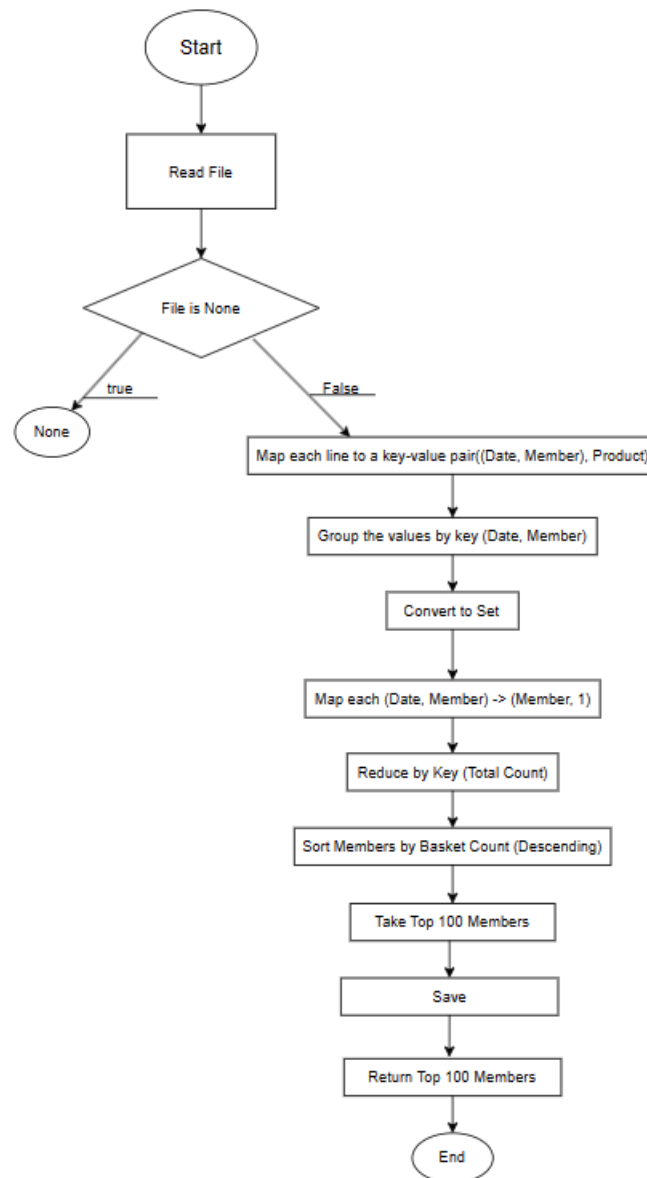


Figure 6: Diagram f3

Function F4

Input: Path to baskets.csv

Output : Print results on the screen and save them to folder **f4**

- ✓ Find the member that bought the largest number of distinct products.
- ✓ Show 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.

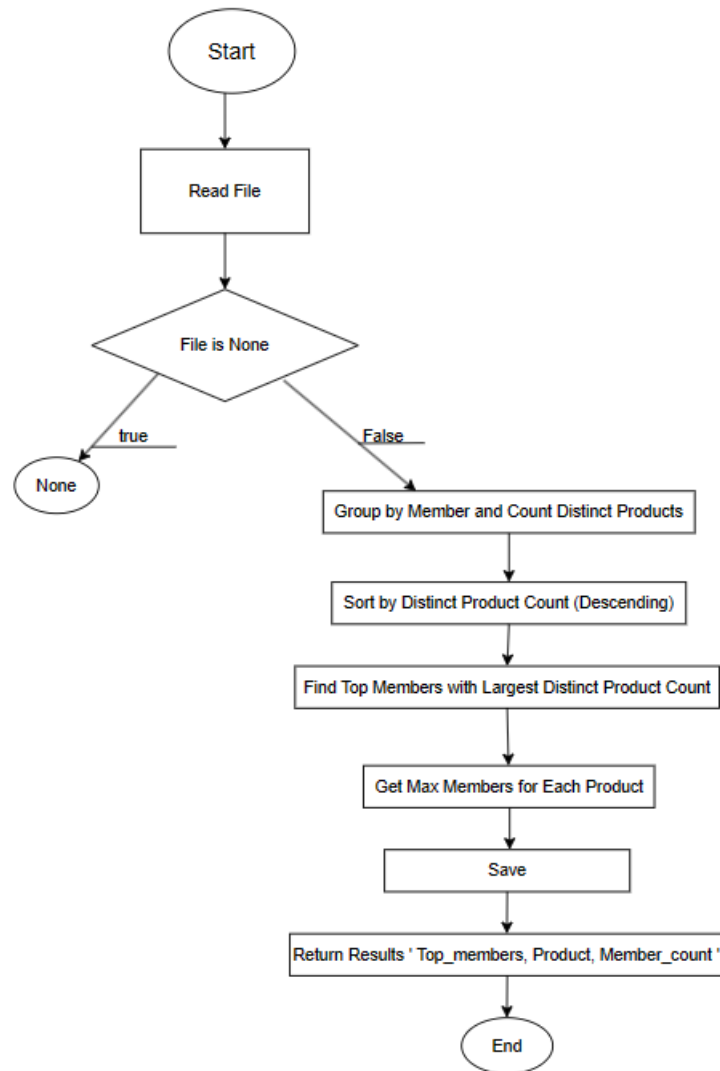


Figure 7: Diagram f4

2.2 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.

Pseudo code:

```
df <- read CSV file into Dataframe
window_spec <- define window specification for partition (
Member_number , year ,
month , day )
Concatenate item descriptions in df within window_spec
Remove duplicate rows in df
Select column in df and sort
Display resulting Dataframe
Total_baskets_each_date <- calc total baskets for each date
pdf <- convert DataFrame ( df ) to Pandas DataFrame
fig , ax <- Create a new figure and axis for the plot
date <- extract Date from Pandas DataFrame
baskets_count <- extract count from Pandas DataFrame
format the date axis
plot the data
display the plot
```

2.3 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**.

Pseudo code:

Initialization: there are two attributes *s* and *c* have been added to the PCY class. These are the support threshold and confidence threshold to the control the finding rules process.

```
class PCY :
    function __init__ ( self , path : str , S : float , C : float
, bucket_size : int = 5000 ) :
        initialize Spark session and context
        set path, S, C, and bucket_size
        read baskets from path
        create item column by exploding the baskets
        calculate total_baskets count
```

find_frequent_items method: count the frequent occurrences of each item in baskets. Then calculate support by dividing the counts of itemsets by the total number of baskets. Finally, filtering items where the support is greater than support threshold

```
function find_frequent_items ( self ) :
    Group items and calculate their counts
    Calculate support for each item
    Filter items based on support threshold
    Return DataFrame containing frequent items
```

generate_pairs method: this method is responsible for generating candidate pairs from frequent items by “Cartesian Product” to frequent items and itself.

```
function _generate_pairs ( self , frequent_items_df ):
```

```

Join frequent_items_df with itself to generate pairs
Select columns to represent pairs
Return DataFrame containing pairs

```

generate_frequent_pairs method: this method is a vital for generating candidate pairs.

```

function generate_frequent_pairs ( self , frequent_items_df ) :
    Generate pairs of frequent items
    Hash pairs and filter based on bucket size and support
threshold
    Join pairs with baskets data to count occurrences
    Return DataFrame containing frequent pairs

```

generation_association_rules method: Generating association rules from the frequent items pairs. This involves creating all possible subsets of a pair and calculating the confidence of each rule based on their frequencies

```

function generate_association_rules ( self , frequent_items_df ,
frequent_pairs ) :
    Generate association rules from frequent pairs
    Calculate support and confidence for each rule
    Filter rules based on support and confidence thresholds
    Return DataFrame containing association rules

```

run method: The rules are then filtered based on specific support and confidence thresholds. Finally, the rules are saved in a CSV file.

```

function run ( self ) :
    Find frequent items
    Generate frequent pairs from frequent items
    Generate association rules from frequent pairs
    Order association rules and save them
    Return frequent pairs and association rules

```


CHAPTER 3 – EVALUATION

3.1 Advantages versus disadvantages

Advantages :

- **Diverse perspectives:** Working in a group brings together individuals with diverse backgrounds, experiences, and ideas. This diversity can lead to innovative solutions and a broader range of insights when tackling problems related to RDD, DataFrame, and PCY algorithm.
- **Collaboration and synergy:** Group work encourages collaboration among team members. They can share their knowledge, skills, and resources, leading to a synergy that enhances the overall quality of the project. Collaborative problem-solving can help overcome challenges more effectively.
- **Reduced workload:** Sharing the workload among team members can reduce the individual burden and prevent burnout. It allows team members to focus on specific aspects of the project, ensuring better attention to detail and overall project quality.

Disadvantages :

- **Coordination and communication challenges:** Managing a group project requires effective coordination and communication. It can be challenging to ensure that all team members are on the same page, have a clear understanding of the project goals, and are working towards them efficiently.
- **Differences in work styles and commitment levels:** Team members may have different work styles, levels of commitment, and priorities. This can lead to conflicts and disagreements within the group, affecting the overall progress and cohesion of the project.

3.2 Task assignments

Full name	Email	Assigned tasks	Completion level
Do Minh Quan	521H0290@student.tdtu.edu.vn	Task 1, 4	100%
Ho Huu An	521H0489@student.tdtu.edu.vn	Task 1, 2, 3, 4	100%
Van Cong Nguyen Phong	521H0287@student.tdtu.edu.vn	Task 1, 2, 4	100%
Nguyen Le Phuoc Tien	521H0514@student.tdtu.edu.vn	Task 4	100%

Table 1: Self-Assessment

3.3 Self- assessment

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

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

Table 2: Task assignments

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). [1]
- [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). [2]
- [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). [3]
- [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.