FP Tree Using Spark

submitted in fulfillment of the requirements

for Major-Project Sem-VII

by

Mr. Karan Singh (170)

Mr. Muhammad Sadiq(169)

Mr. Taher Sadriwala (163)

Mr.Parth Mistry (156)

Supervisor:

Mr. John Kenny

Department of Data Engineering

VIDYA VIKAS EDUCATION TRUST'S
UNIVERSAL COLLEGE OF
ENGINEERING KAMAN, VASAI - 401208
UNIVERSITY OF MUMBAI

2023-2024

Vidya Vikas Education Trust's

Universal College of Engineering, Vasai (E)

Department of Data Engineering

CERTIFICATE

This is to certify that, the Major Project entitled **"FP Tree Using Spark"** is the bonafide work of **Mr. Karan Singh (170), Mr.Muhammad Sadiq (169), Mr.Taher Sadriwala (163) and Mr.Parth Mistry (156)** submitted to the University of Mumbai in fulfilment of the requirement for the Major Project-I Semester VII project work of B.E. Data Engineering at Universal College of Engineering, Vasai,Mumbai at the Department of Data Engineering, in the academic year 2023-2024, Semester – VII .

Mr.John Kenny

Supervisor

Dr. Neeta Patil

Head Of Department

Dr. J.B. Patil

Principal

Major Project-I Report Approval for B. E.

This project report entitled "Searchious" by Mr. Taher Sadriwala, Mr. Karan Singh, Mr. Parth Mistry and Mr. Muhammad Sadiq is approved for the Major Project-I Semester VIII project work of B.E Data Engineering at Universal College of Engineering, Vasai, in the academic year 2023-2024.

Mr. John Kenny
Internal Examiner

22/4/24
External Examiner

Date:

Declaration

I declare that this written submission represents my ideas in my own words, and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated, or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. Karan Singh (170)

Mr. Muhammad Sadiq (169)

Mr.Taher Sadriwala (163)

Mr. Parth Mistry (156)

Date:

Abstract

In recent years, extensive studies have been conducted on high utility itemsets (HUI) mining with wide applications. However, most of them assume that data is stored in centralized databases with a single machine performing the mining tasks. Consequently, existing algorithms cannot be applied to the big data environments, where data is often distributed and too large to be dealt with by a single machine. To address this issue, we propose a new framework for mining high utility itemsets in big data. A novel algorithm named PHUI-Growth (Parallel mining High Utility Itemsets by pattern-Growth) is proposed for parallel mining HUIs on Hadoop platform, which inherits several nice properties of Hadoop, including easy deployment, fault recovery, low communication overheads and high scalability. Moreover, it adopts the MapReduce architecture to partition the whole mining tasks into smaller independent subtasks and uses a Hadoop distributed file system to manage distributed data so that it allows parallel discovery of HUIs from distributed data across multiple commodity computers in a reliable, fault tolerance manner. Experimental results on both synthetic and real datasets show that PHUI-Growth has high performance on large-scale datasets and outperforms state-of-the-art non-parallel type of HUI mining algorithm.Top-k high utility itemset (abbr. Top-k HUI) mining aims at efficiently mining k itemsets having the highest utility without setting the minimum utility thresholds. Although some studies have been conducted on top-k HUI mining recently, they mainly focus on centralized databases and are not scalable for big data environments. To address the above issues, this paper proposes a novel framework for parallel mining of top-k high utility itemsets in big data. Besides, a new algorithm called PKU (Parallel Top-K High Utility Itemset Mining) is proposed for parallel mining of top-k HUIs on Spark in-memory platform.

Keywords-Top-k high utility itemset, In-memory computing, Big data, MapReduce, Spark platform , High utility mining set, Big Data Analytics

Contents

List of Figures

List of Tables

List of abbreviations

1. FI – Frequent Itemset
2. AR – Association Rule
3. CI – Candidate Itemset
4. RDD – Resilient Distributed Dataset
5. IEEE – Institute of Electrical and Electronics Engineers
6. RAM – Random Access Memory
7. GPU – Graphic Processing Unit
8. DFD – Data Flow Diagram
9. UML – Unified Modelling Language

Chapter 1

Introduction

Association rule mining is a fundamental data mining technique with applications in various domains, such as market basket analysis, recommendation systems, and anomaly detection. The FP Tree algorithm is one of the pioneering methods in this field, used to discover interesting associations or patterns within transactional datasets.In this implementation, we will explore the integration of the FP Tree algorithm with Apache Spark.

1.1 Project Idea

In this project, we aim to develop a scalable and efficient implementation of the FP Tree algorithm using Apache Spark, a powerful distributed data processing framework. The FP Tree algorithm is a classic data mining technique used for association rule mining, specifically for analyzing market basket data. By leveraging the distributed capabilities of Spark, we intend to process large transaction datasets to discover valuable associations among items frequently bought together. Market Basket Analysis is a data mining technique used by retailers to discover associations between products that customers tend to buy together. In this project, you will implement the FP Tree algorithm using Apache Spark to perform distributed Market Basket Analysis on a large retail dataset.This project aims to leverage the power of Apache Spark to efficiently implement the FP Tree algorithm for association rule mining. The scalable and parallel nature of Spark will enable us to process large datasets, making it a valuable tool for various data mining applications. The project will culminate in a user-friendly interface for users to interact with and analyze their data effectively.However, it's essential to note that building and maintaining a project based on FP Tree and Apache Spark may also come with some challenges. You will need expertise in both distributed computing with Spark and the FP Tree algorithm itself. Data preprocessing, feature engineering, and result interpretation can be complex and require a deep understanding of your specific use case.

Also, managing the distributed Spark cluster and optimizing your code for performance may be necessary. In summary, implementing the FP Tree algorithm using Apache Spark can have a transformative impact on your data mining and analysis capabilities.

It offers scalability, real-time processing, and significant performance improvements, making it a valuable tool for discovering valuable insights and patterns in large datasets. However, it also requires expertise and careful project management to fully harness its potential. FP Tree algorithm is an influential algorithm that is generally used in the field of data mining & association rule learning. It is used to identify frequent items in a dataset & generate an association-based rule based on the.That's why using this algorithm we want to make a project on big data.

.

Chapter 2

Review of Literature

A literature survey was carried out to find various papers published in international journals such as IEEE etc. related to FP Tree Algorithm using Spark to get the best for the same.

2.1 Existing System

The FP Tree Algorithm is a distributed computing framework that uses Spark to process large datasets efficiently. It involves setting up a Spark cluster, preparing and preprocessing the data,the algorithm for frequent itemset mining, leveraging Spark's parallel processing capabilities, and optimizing the process for efficiency and scalability. The process involves candidate generation, frequent itemset generation, and repeating until no more items are found.Next, you need to process your data to create the transactions and itemsets needed for FP Tree. You may need to perform data cleaning and transformation to suit your analysis.

Parallel mining of itemsets has been implemented in both shared memory and distributed computing architectures. However, scalability remains a major issue with shared memory architecture. In this regard, distributed computing solutions, namely, Apache Hadoop and Spark, not only address the data storage issue through a fault-tolerant Hadoop Distributed File System (HDFS) but also provide utilities that process the data in a real-time manner via MapReduce engine. In this section, a few algorithms for mining frequent itemsets along with high utility itemsets in the aforementioned frameworks are discussed.FP Tree, FP-Growth, and ECLAT are the major centralized algorithms that have been widely adapted for FIM using distributed computing frameworks.With the rapid development of Internet of Things, information transmission and storage technology, the amount of data generated and needed to be mined has explosive growth; we have entered the era of big data. The most basic and prominent feature of big data is the huge amount of data. FIM is a task with high computing load and storage requirements.

If there are n items in the dataset, the size of search space is 2n−1, and the computing load and storage requirements are extremely heavy. When the amount of data is large, distributed algorithms are needed to meet the above challenges. Popular big data distributed computing platforms such as Hadoop and Spark are based on the idea of data localization, which can easily realize efficient, automatic balancing and automatic fault-tolerant distributed mining. Several FIM algorithms based on sliding window using popular big data platforms have been proposed

2.2 Literature Survey

We have examined various research papers in the domain of FP Tree Algorithm using Spark for our project to delve deeper into the details of the various researches conducted in the field of FP Tree Algorithm. Table 2.1 shows a survey of the research paper done for the project.

Table 2.1 – Literature Survey table

Sr. No.	Paper Name	Year of Publication	Author	Publication	Proposed Work	Research Gap
1.	PUC: parallel mining of high-utility itemsets with load balancing on spark	2022	De Gruyter	Journal Of Intelligent Systems	Parallel mining of itemsets has been implemented in both shared memory and distributed computing architectures	This is the research that we will expand on and include into our project. The challenge here is to adapt to a dispersed setting. .

2	A parallel approach for high utility-based frequent pattern mining in a big data environment	2021	Krishna Kumar Mohbey Sunil Kumar	IJCS	The framework of a proposed parallel approach that generates high utility based frequent patterns.	The proposed approach achieves a better result as compared to other high utility based frequent pattern mining.
3	Distributed Mining of Spatial High Utility Itemsets in Very Large Spatiotemporal Databases using Spark In-Memory Computing Architecture	2020	Uday Kiran Rage Minh Dao Incheon Paik	IEEE	Proposes a scalable and fault tolerant distributed algorithm to find all SHUIs in(decentralized)spatiotemporal databases.	There exists no distributed algorithm to find SHUIs In big data, we will evaluate the performance in our project.
4	SWEclat: a frequent itemset	2020	Wen Xiao Juan Hu	TJSC	States that SWEclat algorithm has good accel	In this research the problem of distributed frequent itemset

	mining algorithm over streaming data using Spark Streaming				eration and scalability, and can be used to solve many problems based on frequent itemset mining of streaming data	Fails to mining the streaming data
5	Parallel high utility itemset mining algorithm for big data processing	2019	Krishan Kumar Sethia Dharavath Ramesh Damodar Reddy Edlab	ICCIDS	This research showcases that fast and efficient performance to run big transaction data and distributed across multiple nodes of cluster .	This research fails to establish the running time is lesser than the FHM+. P-FHM+ is much faster than FHM+
6	Parallel High Average	2019	Krishan Kumar Sethi	IEEE	This paper states the	Some methods performed

	-Utility Itemset Mining Using Better Search Space Division Approach		Dharavath Ramesh M. Sreenu		Spark-based distributed algorithm that model the HAUI-Miner to work in parallel on a multi node cluster.	that PHAUIM outperforms the HAUI-Miner with a huge margin
7	Efficient Parallel Algorithm for Mining High Utility Patterns Based on Spark	2018	Junqiang Liu Rong Zhao Xiangcai Yang Yong Zhang Xiaoning Jiang	IEEE	This paper Presents Pattern mining as a data mining technology that aims to help people to obtain valuable patterns from a large number of data.	The existing utility pattern mining algorithms, no matter two phase ones or single phase ones, are only suitable for running on a single machine

8	Fuzzy High-Utility Pattern Mining in Parallel and Distributed Hadoop Framework	2018	Jimmy Ming-Tai Wu Gautam Srivastava Min Wei Unil Yun Jerry ChunWei Li	ISIJ	This section proposes an itemset fuzzy upper bound, an algorithm for constructing a searching graph, and pruning strategies for the developed EFUPM	This paper presents an efficient high fuzzy utility itemset mining (EFUPM) algorithm for revealing the high fuzzy utility patterns.

In the paper presented by De Gruyter , In this study, a Spark-based PUC (Parallel Utility Computation) algorithm for mining HUIs has been proposed. The algorithm is a parallel adaptation of the SPUC algorithm. PUC considers TWU values of items as a measure of transaction similarity. To this end, by grouping the items using Jenks Natural Breaks algorithm, mining performance improved as the number of local projections reduced, a major factor that contributes to the shuffling cost. Experimental evaluation also demonstrated that grouping items based on JNB clustering of TWU values outperformed in comparison to the same number of groups generated after sorting TWU in either ascending order or descending order [1].

In the paper presented by Krishna Kumar Mohbey ,Sunil Kumar, In this work, we proposed a novel approach to generate high utility based frequent patterns. The generated patterns can be further used in different areas for recommendation and decision making. Besides, this approach can handle a huge amount of data and recommend various services to individuals, business companies, and government agencies. This approach can be utilized in various fields such as education, healthcare, banking, finance, communication and so on. Other than it, there is a wide range of areas, where high utility patterns will be used for prediction, recommendations as well as decision making [2].

In the paper presented by Uday Kiran Rage,Minh Dao,Incheon Paik, they have proposed a scalable and fault tolerant distributed algorithm to find all SHUIs that may exist in a very large spatiotemporal database. A novel pruning technique and a new load balancing solution were presented to effectively reduce the search space and distribute the load across many computing nodes [3].

In the paper presented by Wen Xiao, Juan Hu In this paper, we study the problem of distributed frequent itemset mining over streaming data and propose a distributed algorithm named SWEclat based on Spark Streaming which is a popular streaming data processing platform. The SWEclat algorithm uses sliding window technology to access partial data in the streaming data. Because the vertical data structure used by the Eclat algorithm has better insertion and deletion performance, and can be divided into independent search space and conditional database, it is used for storing data in the current window [4].

In the paper presented by Krishan Kumar Sethia, Dharavath Ramesh , Damodar Reddy Edlab A novel HUI mining algorithm called parallel FHM+ (P-FHM+) is proposed in this paper. P-FHM+ models the state-of-the-art algorithm FHM+ to work in the distributed environment of Apache Spark. FHM+ introduces a new concept for upper length reduction to discover high utility itemsets with length constraints. Traditional HUI mining algorithms are not appropriate to work on big data. A stand alone system is replaced by a distributed system to store and process big data in an efficient way [5].

In the paper presented by Krishan Kumar Sethi, Dharavath Ramesh, M. Sreenu, this paper presents a detailed study about the pattern mining approaches such as HUI (High utility itemset) mining and HAUI (High-average utility itemset) mining. With the introduction to big data, pattern mining has acquired many challenges such as searching for high average-utility item sets from a big transaction dataset. Huge number of candidate itemsets are generated while HAUI searching which requires massive computing resources and memory. A stand alone system is not suitable to process big data in an efficient manner [6].

In the paper presented by Junqiang Liu, Rong Zhao, Xiangcai Yang, Yong Zhang, Xiaoning Jiang , they have designed a parallel high utility pattern mining algorithm PhpMR based on MapReduce for comparison as there is no competitor so far. The overall framework of PhpMR is the same as that of Phps, the difference is that the three phases of the algorithm are implemented as three MapReduce Jobs [7].

In the paper presented by Jimmy Ming-Tai Wu, Gautam Srivastava, Min Wei Unil ,Yun Jerry ChunWei Li, With the rapid growth of commercial datasets, it is necessary to develop an effective mining algorithm to reveal knowledge for decision making. However, the traditional mining algorithms were limited in the ability to handle the huge dataset. In this paper, we designed two algorithms named EFUPM and HFUPM for mining high fuzzy utility patterns. Two new upper-bounds are 650 also estimated and developed in this paper to reduce the search space of the unpromising candidates [8].

2.3 Problem Statement and Objective

To Create and develop a distributed scalable FP Tree algorithm using Apache Spark to quickly find popular itemsets within huge transactional datasets.The objective is to offer a distributed and parallelized association rule mining method that enables the investigation of market basket data,consumer behavior, or any other domain.

2.4 Project Scope

A common approach used for frequent itemset mining and association rule discovery is the FP Tree algorithm. When utilizing Apache Spark to implement the FP Tree method, the project scope often entails creating a distributed and parallelized version of the algorithm that is capable of effectively handling big datasets.

.

Chapter 3

Proposed System

This chapter includes a brief description of the proposed system and explores the different modules involved along with the various models through which this system is understood and represented.

3.1 Analysis/Framework/ Algorithm

The FP Tree algorithm is a powerful tool for solving complex problems by analyzing and cleaning data. It involves several steps, including data preprocessing, data transformation, data encoding, and data splitting. The algorithm's requirements are understood, and the programming language and technology stack chosen, such as Apache Spark, are chosen.

The framework design is defined, including directories, files, and modules, to enhance maintainability and scalability. The code for the algorithm is written, utilizing efficient data structures and algorithms to reduce computational complexity. Performance optimization is also optimized for speed and memory efficiency, using techniques like pruning, in-memory data structures, and parallel processing capabilities.Testing is conducted to ensure the correctness of the implementation and validate its performance with various datasets. Documentation is created, detailing the code, parameters, dependencies, dataset, preprocessing steps, and assumptions made during the project. Visualization tools and reports can be used to present results, making them more accessible to stakeholders.

Testing is conducted to ensure the correctness of the implementation and validate its performance with various datasets. Documentation is created, detailing the code, parameters, dependencies, dataset, preprocessing steps, and assumptions made during the project.

Integration is possible, integrating the algorithm with other systems or tools depending on the project's requirements. Security and privacy concerns should be addressed, and data handling should be secure. Maintenance and updates are planned to keep the implementation up-to-date with changes in the technology stack. Scaling is necessary to handle larger datasets or increased workloads. Performance monitoring and optimization are implemented for real-time tracking and issue identification. User training and support are provided if the algorithm is intended for non-technical users.Flexibility in adapting the framework to meet specific project requirements is crucial for a successful implementation.

Step 1 : Generate Candidate Itemsets (Ck) Ck = FP TreeGen(Fk-1)
Step 2 : Calculate Support for Candidate Itemsets (Supp(Ck)) Supp(Ck) = count({transaction | transaction contains Ck}) / Total number of transactions
Step 3 : Prune Infrequent Candidate Itemsets (Lk) Lk = {Ck | Supp(Ck) >= minSup}
Step 4 : Generate Association Rules (A) A = {Association rule |generateAssociationRule(Lk)}
Step 5 : Repeat or Terminate

3.2 System Requirements

This section will provide the user the required specification of the hardware and software components on which the proposed system is to be implemented.

3.2.1 Hardware Requirements

This subsection will provide the minimum requirements that must be fulfilled by the hardware components. The hardware requirements are as follows: -

1.CPU

2.MEMORY (RAM)

3.Storage Graphic Processing Unit (GPU)

4.Operating System

3.2.2 Software Requirements

This subsection will provide the versions of software applications that must be installed. The software requirements are as follows: -

- Apache Spark
- Hadoop HDFS
- Programming Language (Python)
- Spark Libraries
- Cluster Setup
- Dataset

3.3 Design Details

In design details, we analyze the System Architecture and System Modules in detail. We study the flow and process of the entire project in order to develop the project in an orderly and systematic manner.

3.3.1 System Architecture

The System Architecture of Implementation of FP Tree Algorithm using Apache Spark Illustrates that The process begins with 'Utility details' and 'minutility'. It then processes a 'Dataset' using 'Preprocessing activities'. The preprocessed data is stored in 'HDFS/RDD format' before 'generate node data' takes place. Multiple repetitions of the stages 'Data node' and 'mining' suggest these steps are part of a loop in the data mining process. The mining process results in 'frequent patterns'. The entire process culminates in the formation of a 'Decisive/ Recommender system'. In figure 3.1, we show the detailed system architecture.

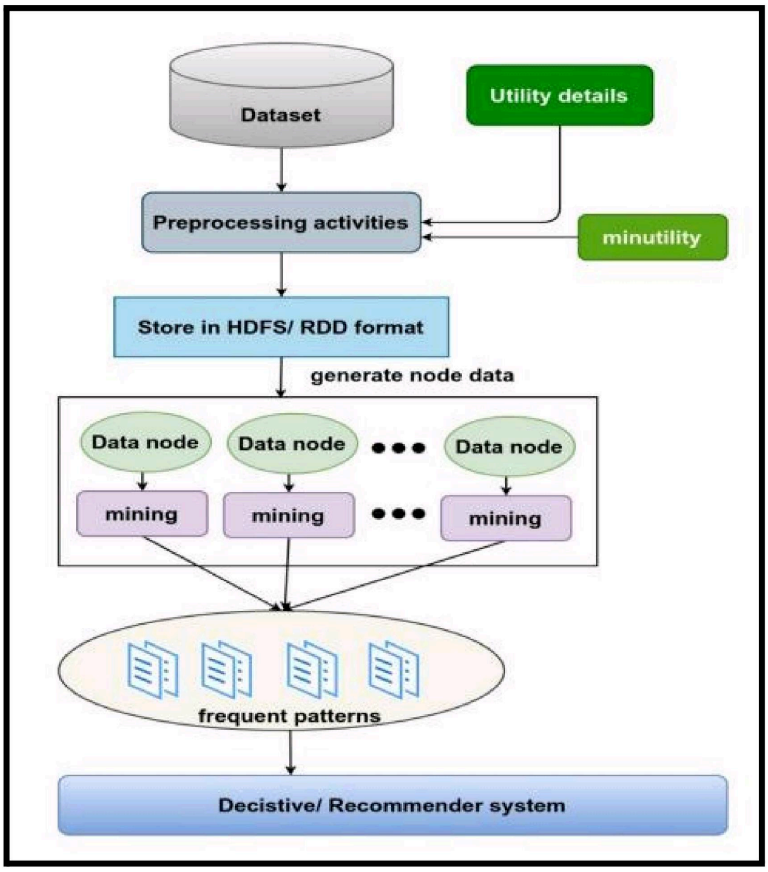


Figure. 3.1 – System Architecture

3.3.2 Details of Modules

FP Tree uses Sparktends to various modules in order to facilitate the frequent itemsets from big datasets .

The modules are:

1. Spark Job
2. Preprocessing
3. FP TreeAlgorithm:

A. Spark Job

The Spark job module is the entry point for the entire system . It represents the main Spark application that orchestrates the entire process.It's responsible for initializing Spark, invoking the data loading and preprocessing, running the FP Tree algorithm, and managing the output.

B. Preprocessing

The Preprocessing module is responsible for preparing the input data for the FP Tree algorithm. It involves loading and processing the transaction data. This method loads transaction data from a file, where filePath is the path to the transaction dataset. It returns an RDD (Resilient Distributed Dataset) of strings, where each string represents a transaction.

C. FP Tree Algorithm

The FP TreeAlgorithm module is where the core FP Tree algorithm is implemented. It discovers frequent item sets from the input data.FP Tree(itemSets: RDD[ItemSet], minSupport: Double): RDD[ItemSet]: This method takes a set of candidate item sets (often the output of the preprocessing step) and the minimum support threshold as input. It then applies the FP Tree algorithm to discover frequent item sets. The result is an RDD of frequent item sets.This module typically involves multiple iterations, pruning, and filtering to identify item sets that meet the minimum support threshold.

3.4 Data Model and Description

Data Model describes the relationship and association among data which includes Entity Relationship Model.

3.4.1 Entity Relationship Model

Figure 3.4 shows the Entity Relationship Diagram of the proposed system. Entity Relationship diagram is a data modeling technique that graphically illustrates an information system’s entities and the relationships between those entities.

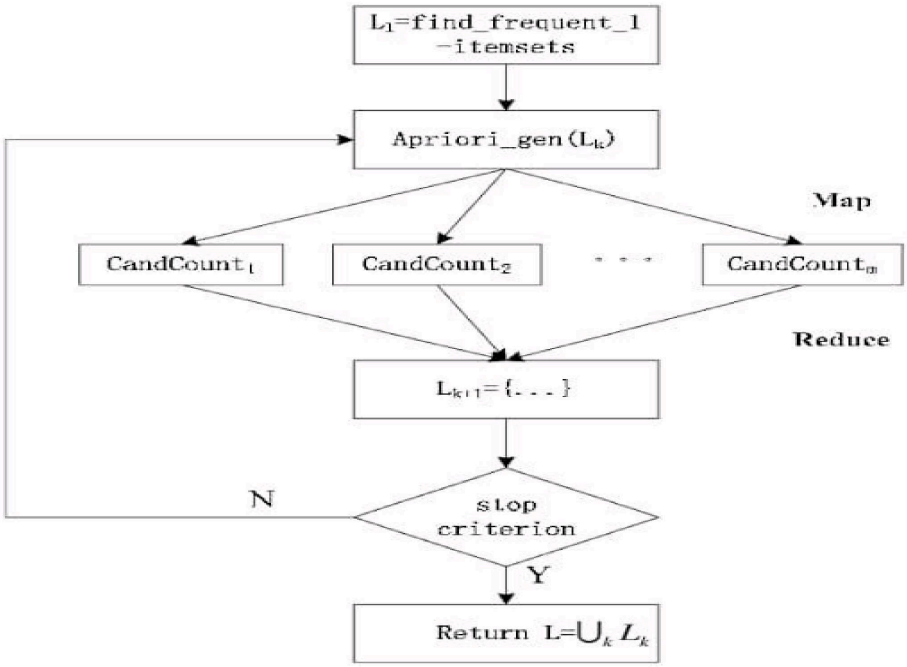


Figure 3.2 - Entity Relationship Diagram

3.5 Fundamental Model

Fundamental model of the project gives an overall idea about the project. How the entities are related to each other, what are the attributes of the entities, how the data flows between the entities is shown by the fundamental model.

3.5.1 Data Flow Model

Data Flow Diagram (DFD) shows graphical representation of the flow of data through an information system, modeling its process aspects. It includes data inputs and outputs, data stores, and the various sub processes the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

DFD LEVEL 0

Figure 3.5 denotes the Level 0 Data Flow Diagram of the proposed system. It is also known as the Context Diagram. This is the most basic representation of the system. It shows a data system as a whole and emphasizes the way it interacts with external entities. It is a complex

representation of the entire system. It displays the most abstract form of a system It gives a quick idea about the data flow inside the system. There is only one visible process that represents the functions of a complete system. The system for simplification is divided by three entities that make up the level 0 DFD .

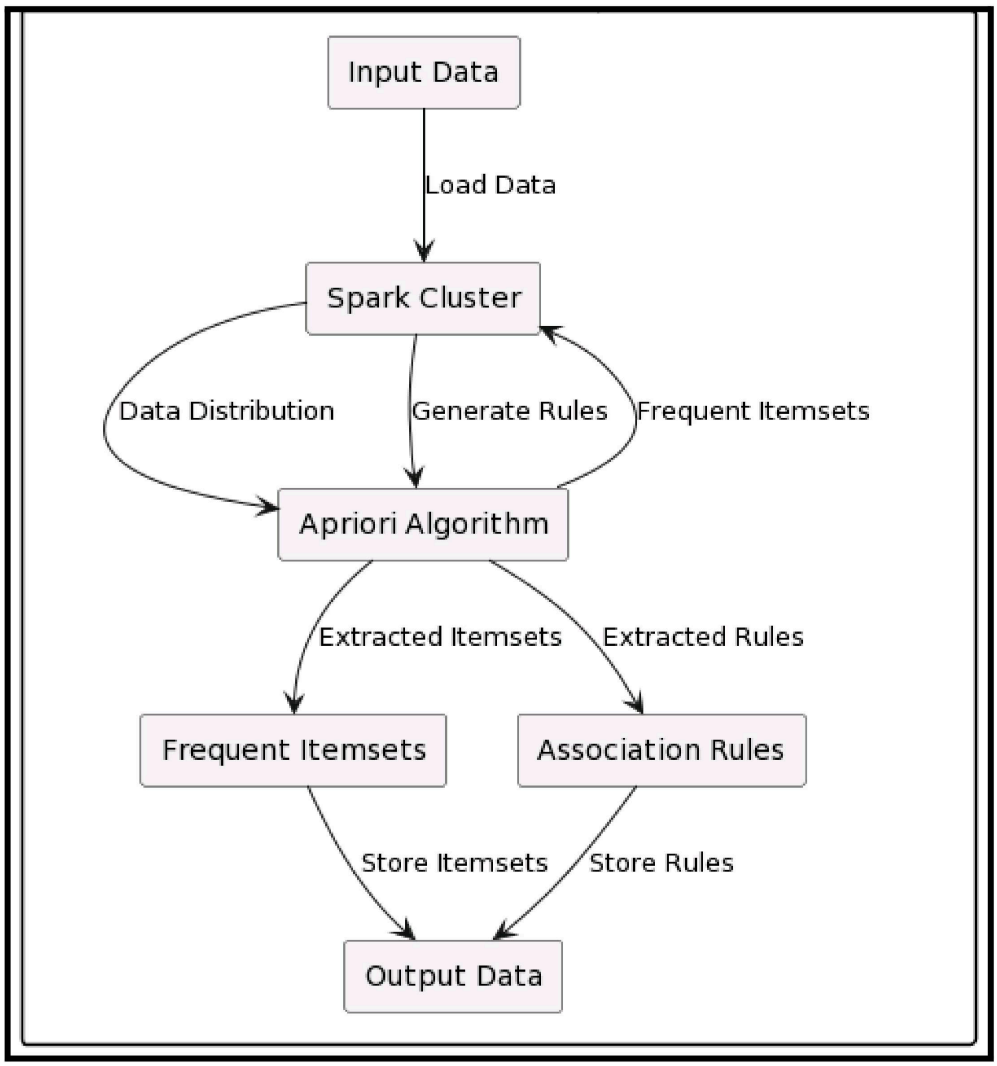


Figure 3.3 – DFD Level 0

DFD Level 1

Figure 3.6 shows the Level 1 Data Flow Diagram of the proposed system. It is exactly the same as the Level 0 DFD, but much simplified. The Level 1 DFD shows how the system is divided into subsystems i.e. subprocesses, each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the

system as a whole. It breaks down the main processes and subprocesses that can then be analyzed and improved on a more intimate level. In this Level 1 diagram, we provide more detail on the processes within the "FP Tree Algorithm" component.

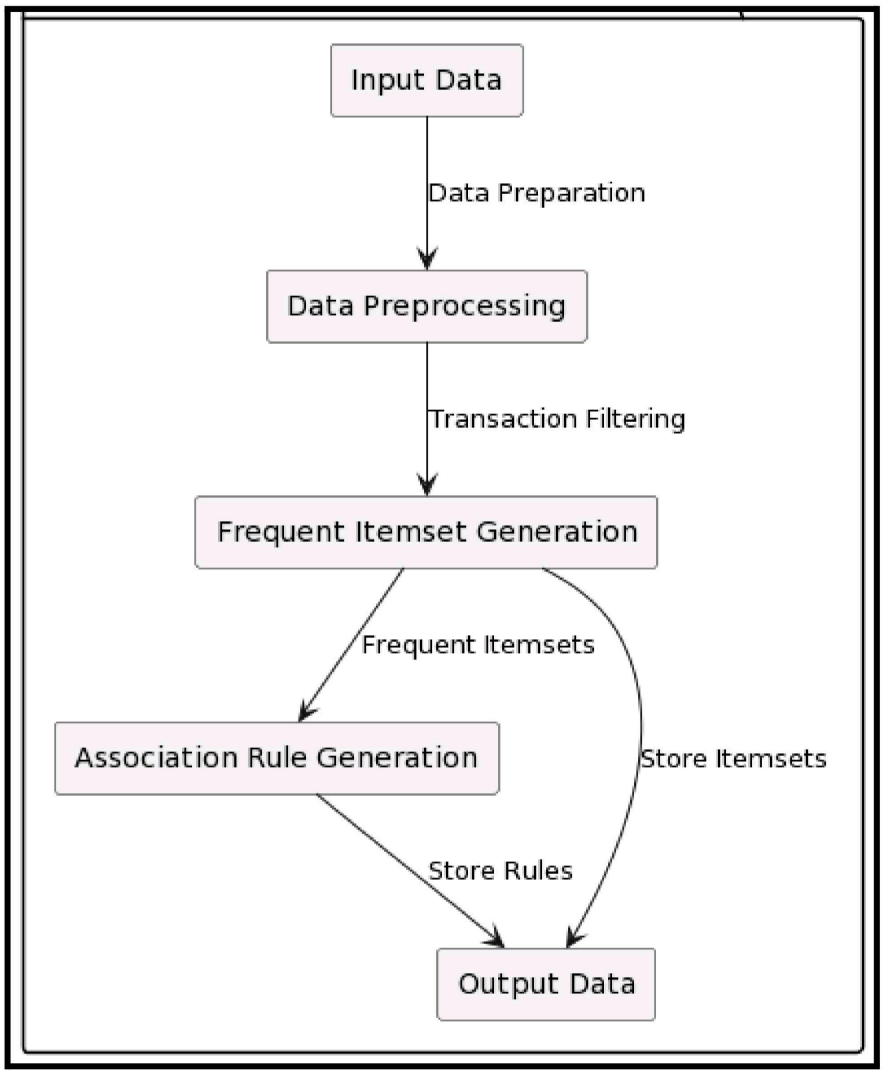


Figure 3.4 – DFD Level 1

DFD LEVEL 2

Figure 3.7 shows the Level 2 Data Flow Diagram of the proposed system. It is exactly the same as the Level 1 DFD, but much simplified.

The Level 2 DFD shows how the system is divided into sub- sub systems i.e. subprocesses, each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole.The level 2 Data Flow Diagram Focuses on data cleaning, transformation, candidate generation, support counting, and rule generation.

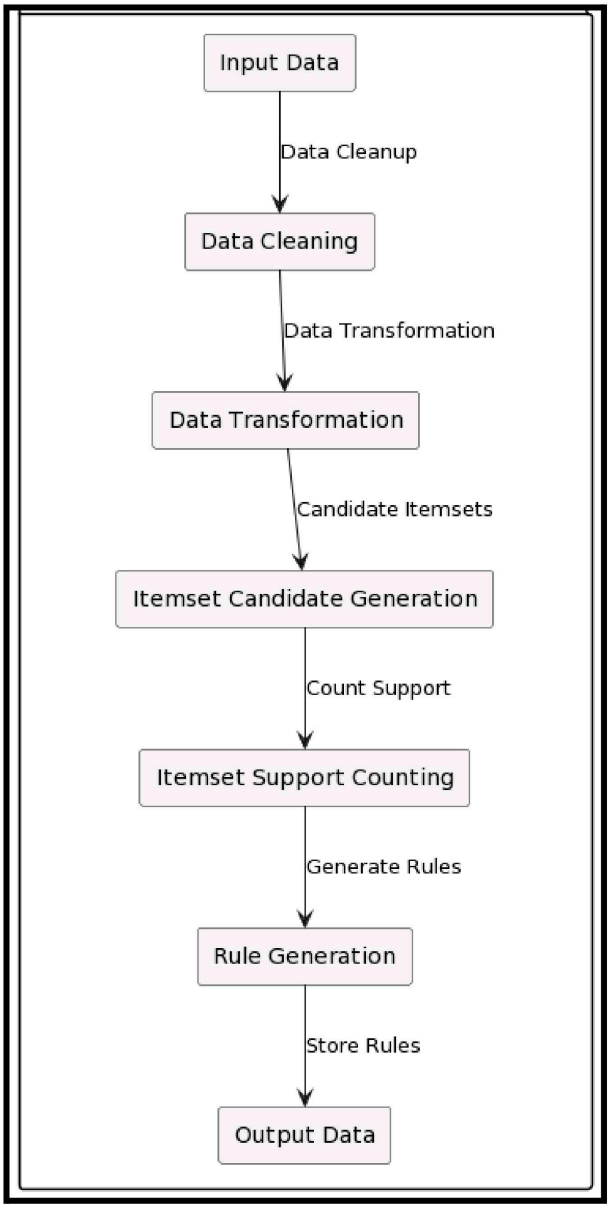


Figure 3.5 – DFD Level 2

3.6 UML (Unified Modelling Language) Diagram

The Unified Modelling Language is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. We have prepared and designed the UML diagrams of – Use Case, Activity, Component, Deployment and Sequence Diagrams.

3.6.1 Use Case Diagram

Figure 3.8 denotes the Use Case Diagram of the proposed system. It shows the user's interaction with the systems. The purpose of a use case diagram in Unified Modelling Language (UML) is to demonstrate the different ways that a user might interact with a system.

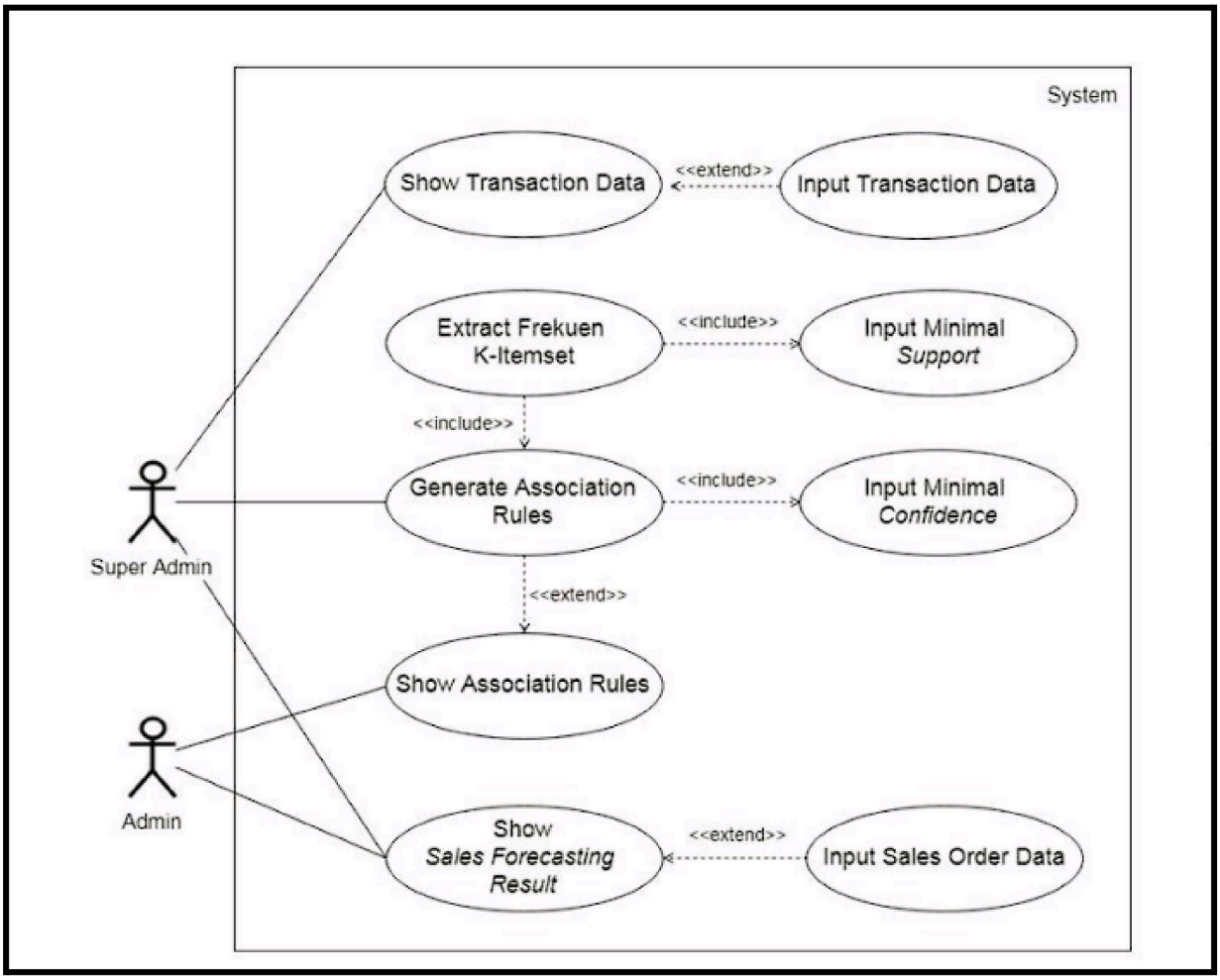


Figure 3.6 – Use Case Diagram

3.6.2 Activity Diagram

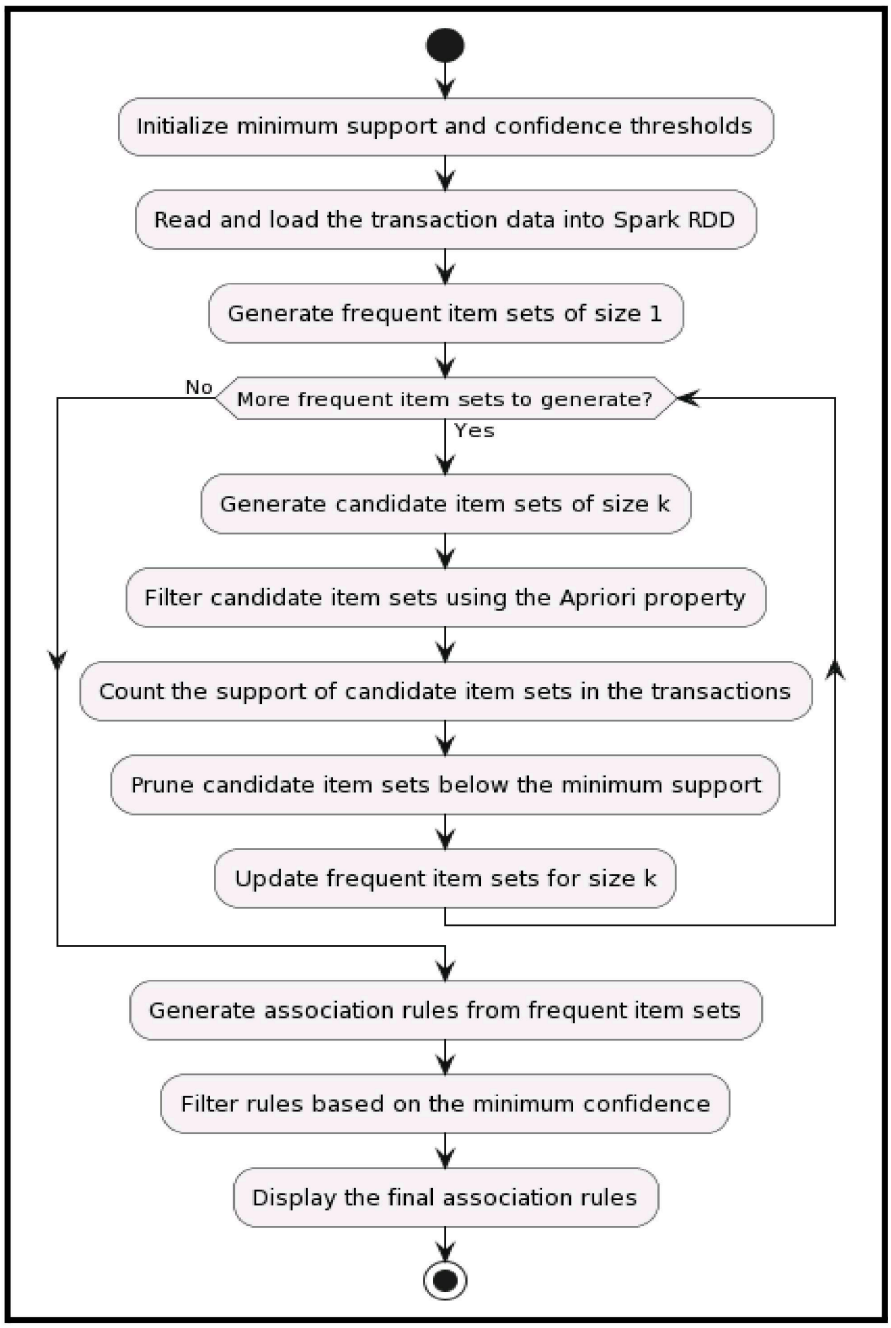


Figure 3.7 – Activity Diagram

In figure 3.9, we can observe the activity diagram. The FP Tree algorithm uses Apache Spark to find frequent item sets in large datasets. It starts with frequent item sets of size 1, generates candidate sets of size k, filters them using the FP Tree property, counts their support, prunes non-supporting sets, updates frequent item sets, generates association rules, filters them based on confidence thresholds, and displays the final rules.

3.6.3 Sequence Diagram

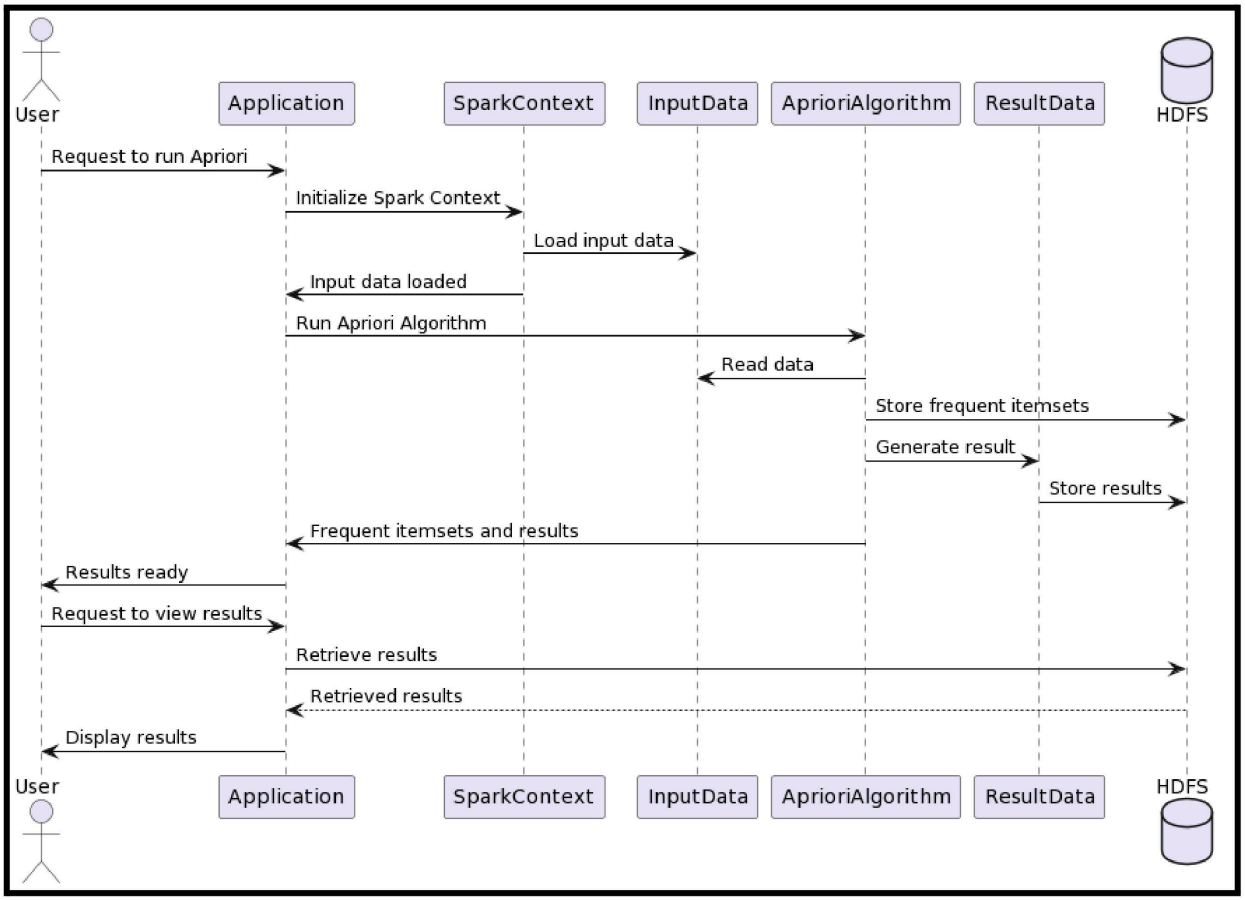


Figure 3.8 – Sequence Diagram

In figure 3.10, we can observe that the sequence diagram demonstrates the high-level flow of interactions between the user, the application, Apache Spark components (SparkContext), the FP Tree Algorithm, data storage (HDFS), and the presentation of results.

3.6.4 Component Diagram

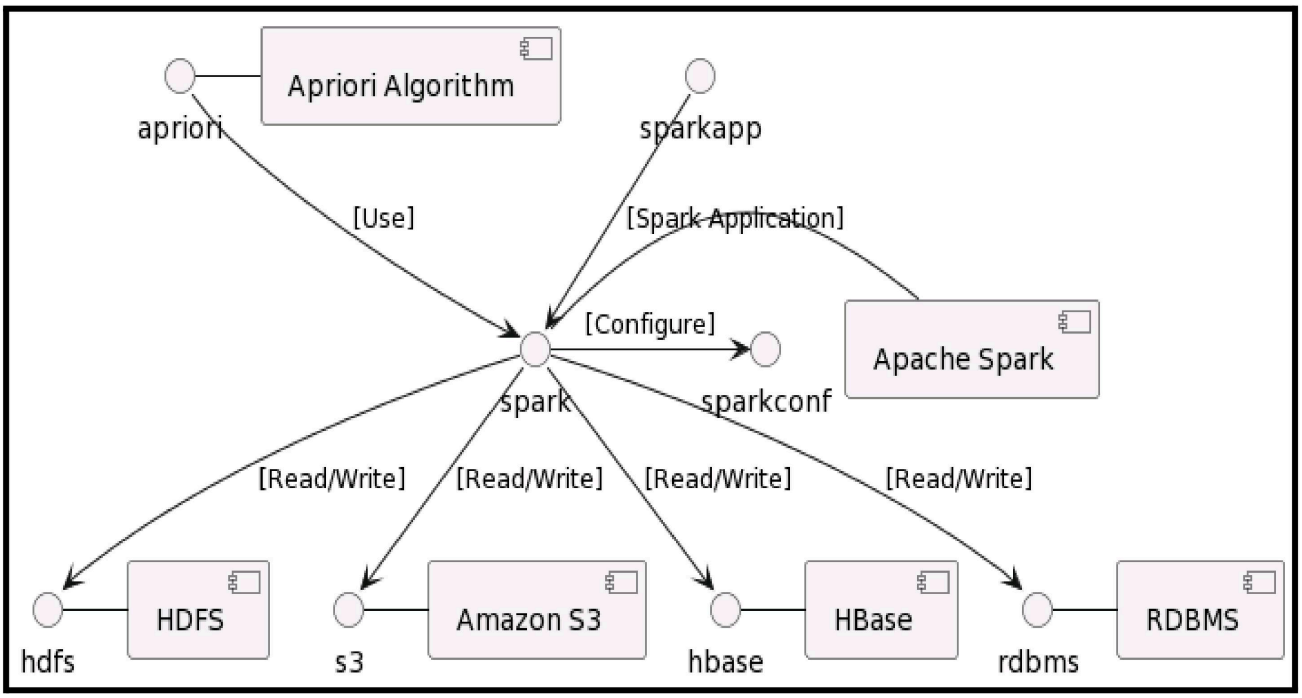


Figure 3.9- Component Diagram

In Unified Modeling Language, a component diagram depicts how components are wired together to form larger components or software systems. So, in simple terms, Apache Spark is the main engine doing the work, and it can grab data from different storage options and process it using the FP Tree Algorithm. This diagram helps us understand the different parts of the system and how they connect to each other.

3.7 Methodology

FP Tree algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. In other words, we can say that the FP Tree algorithm is an association rule learning that analyzes that people who bought product A also bought product B.

The primary objective of the FP Tree algorithm is to create the association rule between different objects. The association rule describes how two or more objects are related to one another. FP Tree algorithm is also called frequent pattern mining. Generally, you operate the FP Tree algorithm on a database that consists of a huge number of transactions.

- Data Preprocessing:

  Load your transaction data into a Spark DataFrame or RDD.

  Prepare your data so that it's in a format suitable for mining frequent itemsets. Typically, this involves representing each transaction as a list of items.

- Map-Reduce Paradigm:

  To effectively complete the necessary FP Tree processes, apply Spark's map-reduce paradigm. Candidate creation and support counting are two tasks that the FP Tree algorithm might divide across Spark's worker nodes.

- Candidate Generation:

  Use the self-join step to construct possible candidate sets and generate candidate itemsets in an efficient manner. You can efficiently parallelize this process with Spark.

- Support Counting:

Count the support of itemsets in your transaction data by utilizing Spark's parallel processing .

- Iteration:

Until you have the final frequent itemsets that satisfy the designated support threshold, repeat the candidate creation, support counting, and cutting procedures repeatedly.

- Result Collection:

As the last product, gather and display the frequently occurring itemsets and related support values.

• Methodology / Approach

Step 1: Load the transaction data into a list of sets

Step 2: Initialize variables

Step 3: Generate frequent 1-itemsets

Step 4: Candidate Generation

```
candidate_itemsets = generate_candidate_itemsets(frequent_itemsets[k - 2])
```

Step 5: Support Counting

```
for transaction in transactions:
for candidate in candidate_itemsets:
 if is_subset(candidate, transaction):
 candidate.count += 1
```

Step 6: Pruning

```
frequent_k_itemsets = prune_infrequent(candidate_itemsets, min_support)
```

Step 7: Add frequent k-itemsets to the list of frequent itemsets

```
frequent_itemsets.append(frequent_k_itemsets)
```

Step 8:Output the frequent itemsets and their support values

```
output_frequent_itemsets(frequent_itemsets)
```

• Technologies Used

The foundational technology for distributed data processing is Apache Spark. To efficiently execute the FP Tree technique on big data, it offers a framework for processing massive datasets in parallel.

Following are list of libraries used:

1. Apache Spark
2. Spark MLlib
3. NumPy
4. Pandas
5. DataFrames & Dataset
6. Cluster Management Tools

Chapter 4

Result and Discussion

This chapter includes the snapshots of the actual outputs that were seen by the user and this chapter also contains the results of the proposed system.

4.1 Proposed System Result

The result of implementing the FP Tree algorithm in a proposed system using Apache Spark would typically be a set of frequent itemsets that meet a specified minimum support threshold. These frequent itemsets represent patterns of items that frequently co-occur in the transaction data.

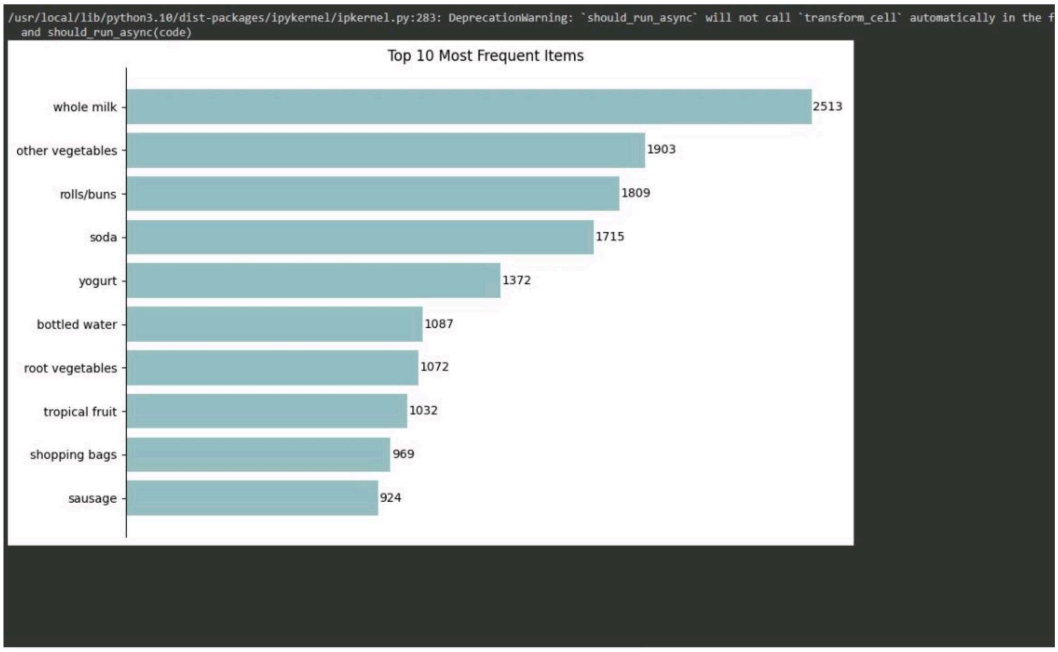


Figure 4.1.1 – Most Frequent Items

Figure 4.2 contains the screenshot of the Top 5 Most Frequent Standalone Items present in the dataset.

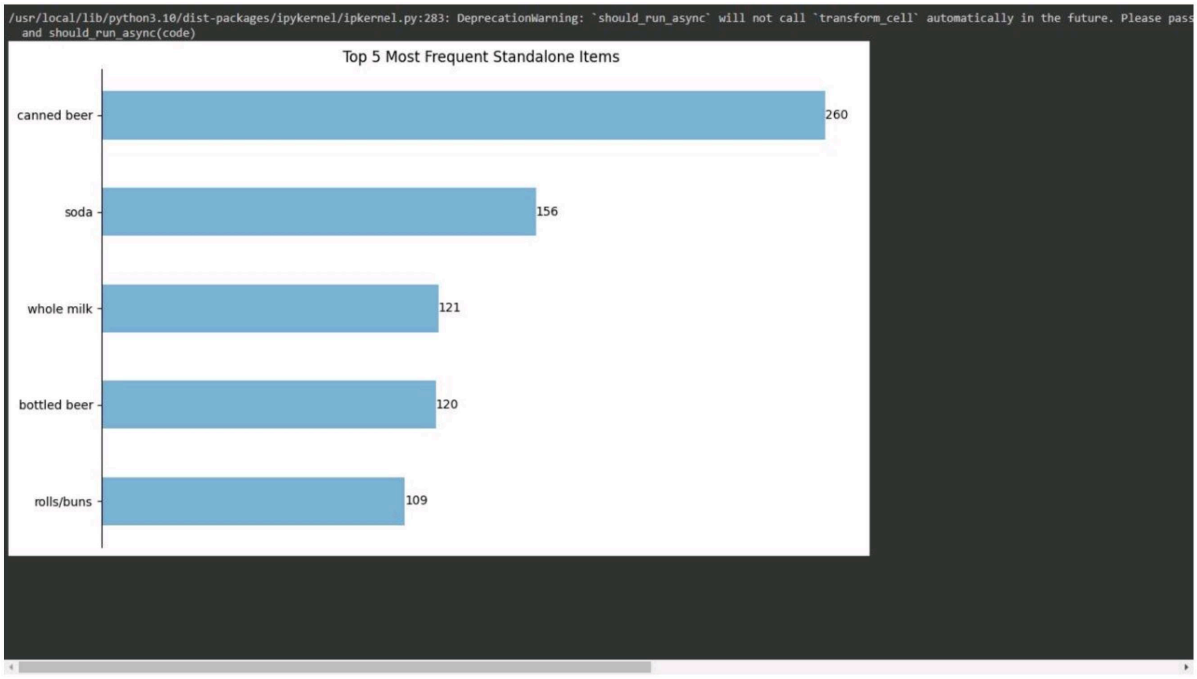
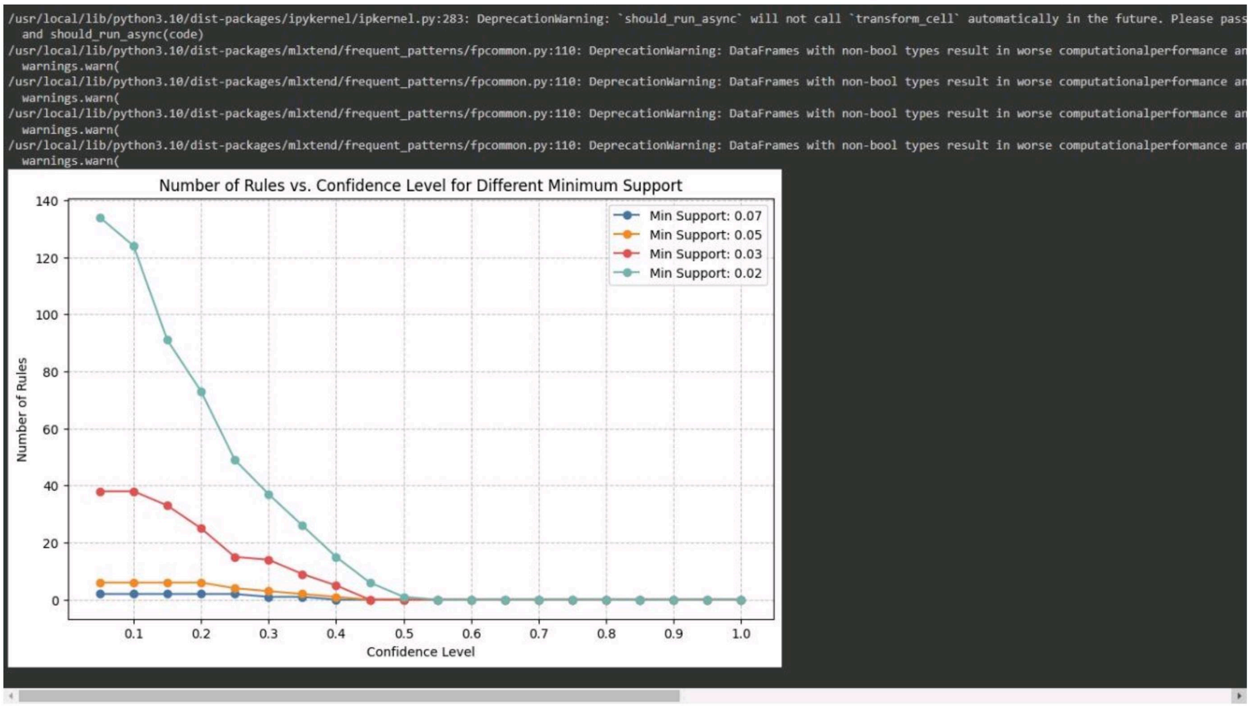


Figure 4.1.2 – Frequent Standalone Items



.Figure 4.1.3 – Number of Rules vs Confidence Level

Snapshot of Project

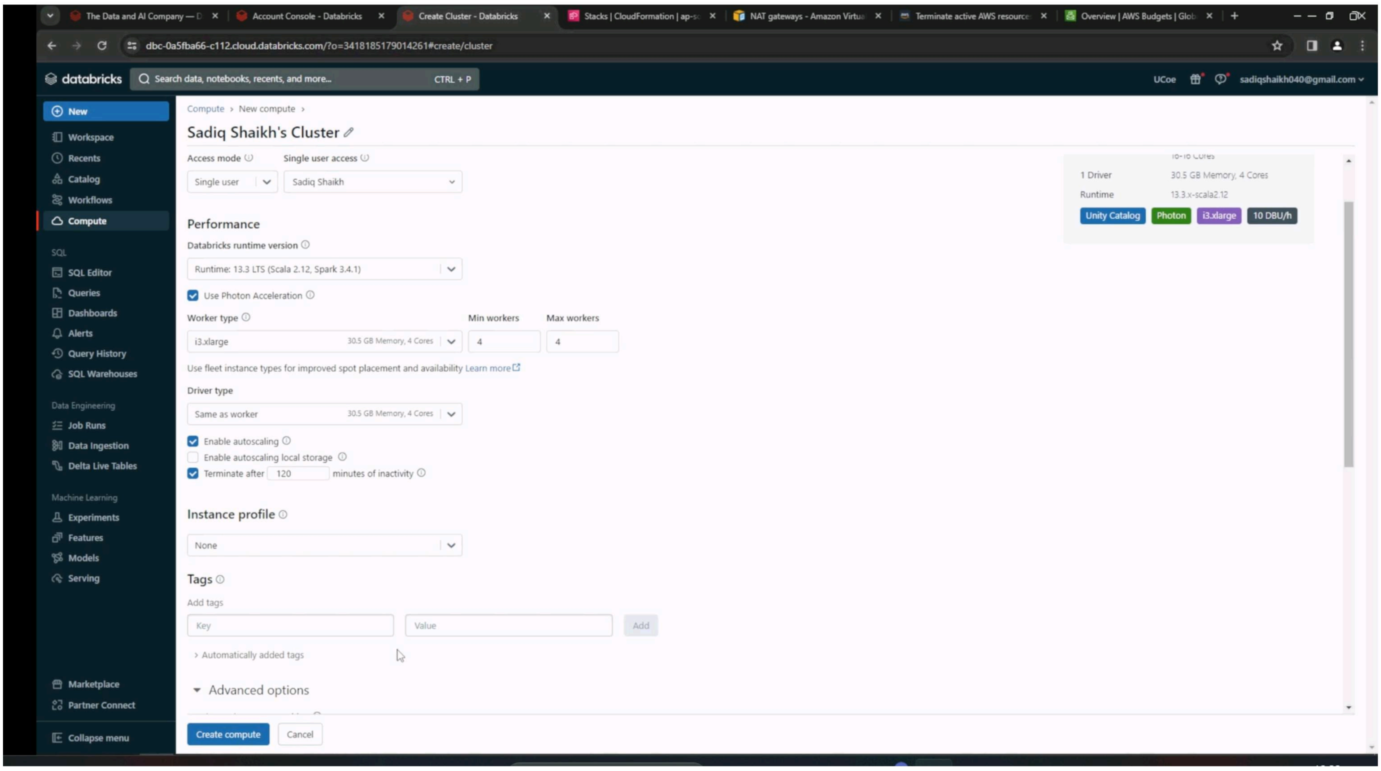


Fig 4.1.4 Creating Cluster

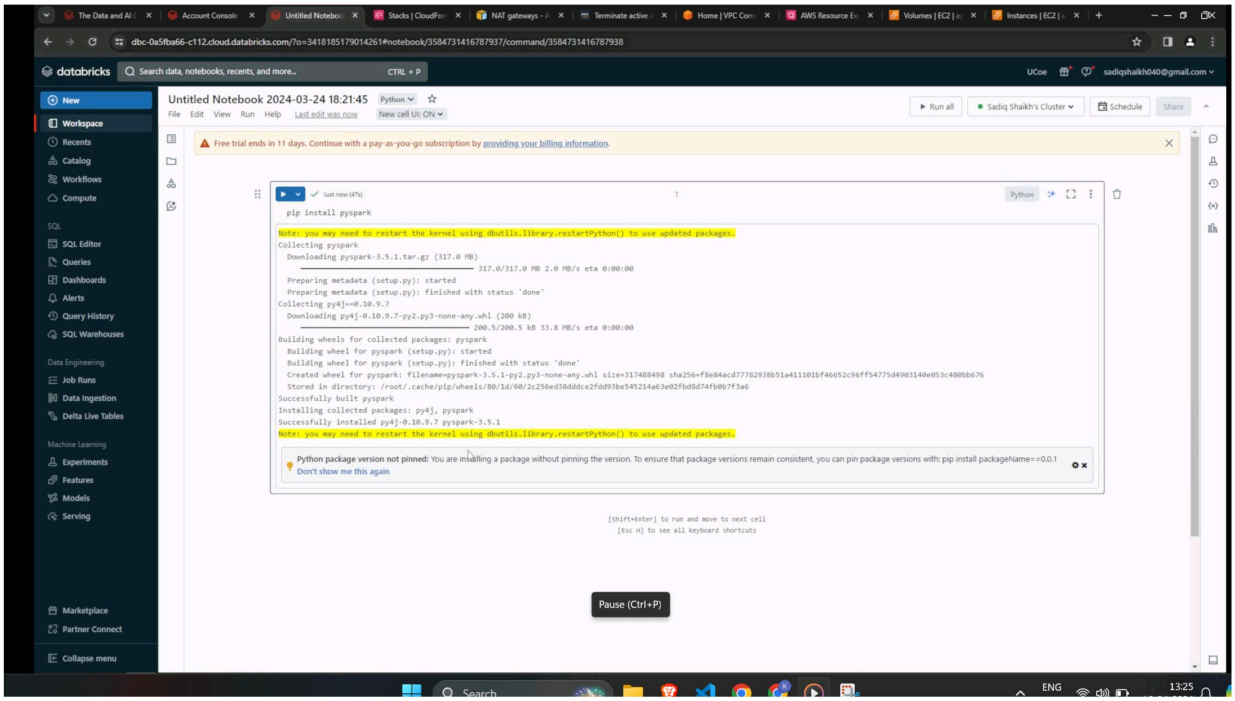


Fig 4.1.5 Installing Libraries

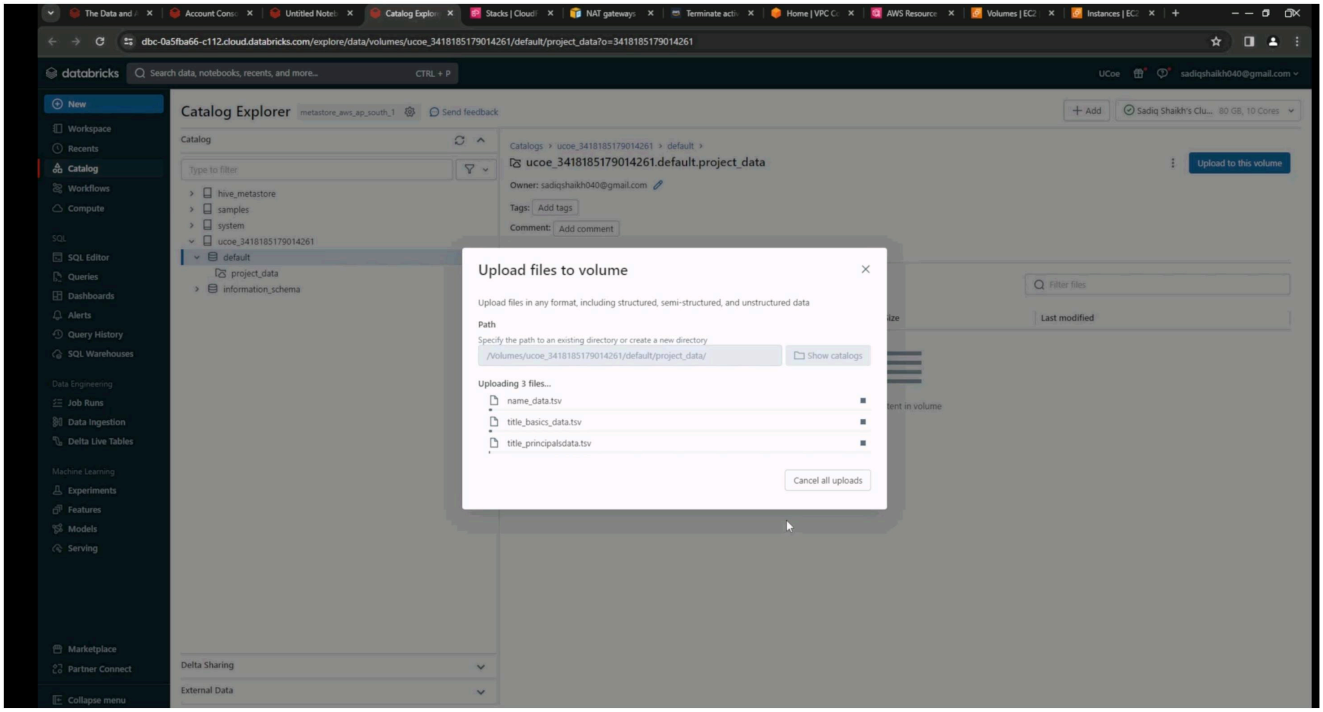


Fig 4.1.6 Adding csv & tsv file

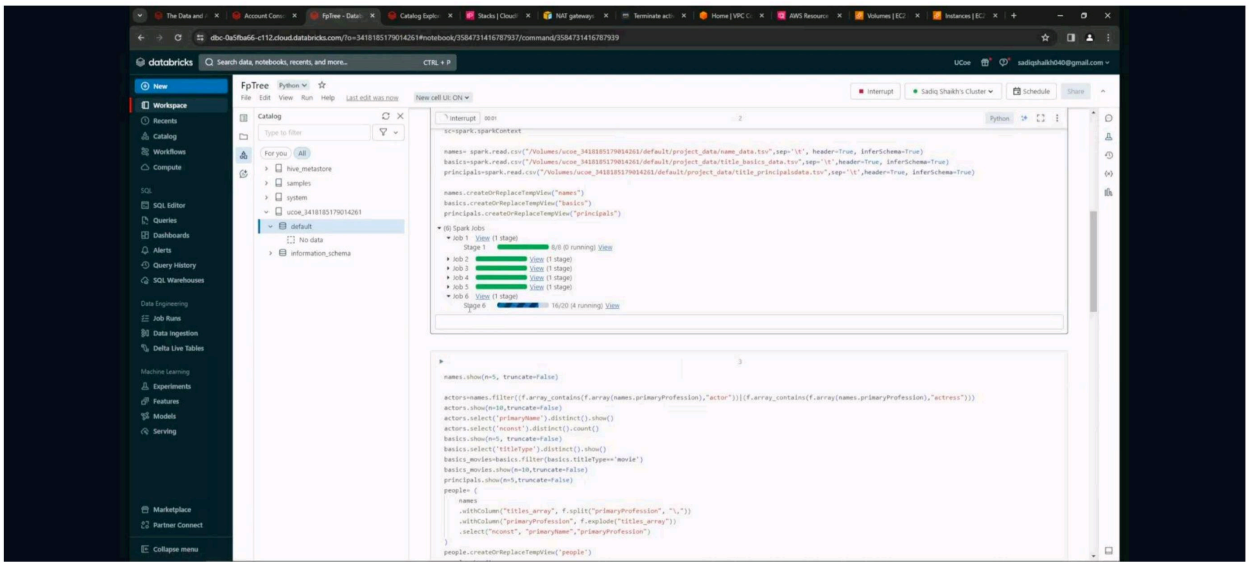


Fig 4.1.7

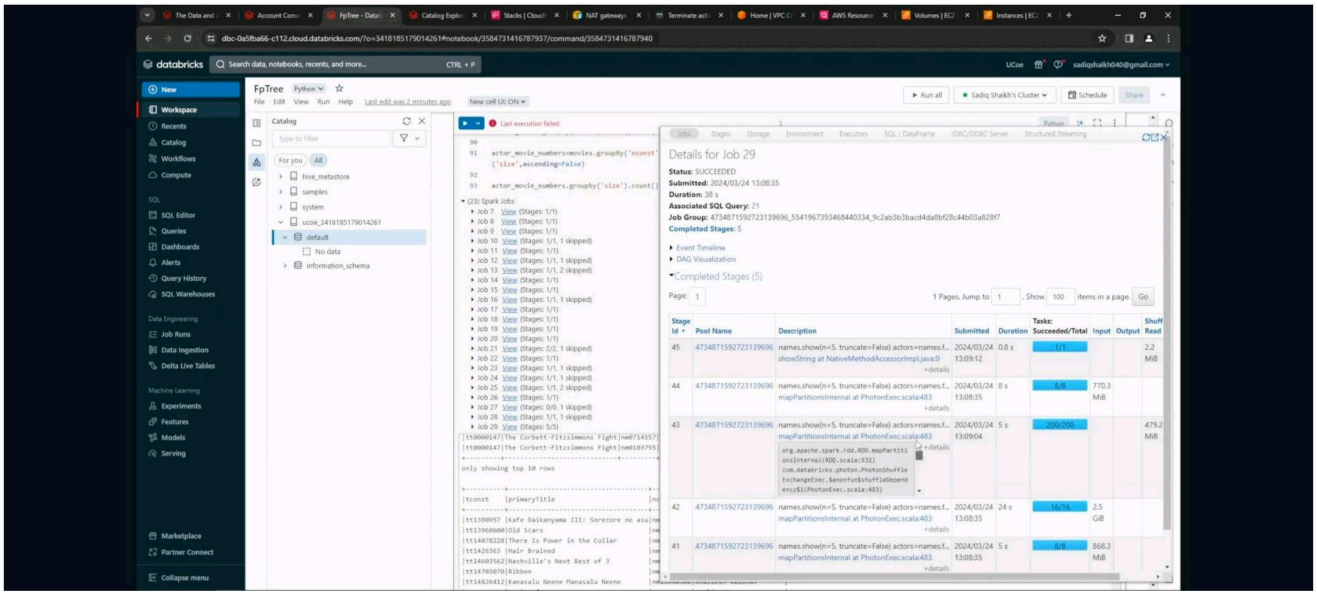


Fig 4.1.8 Final Output of Project

4.2 Proposed system versus existing system

The below illustrated table explains the key points of difference between the existing system and our proposed system i.e, FP Tree Using Spark. The parameter differentiaton are – Scalability, Performance, Efficiency, Real Time Processing, Ease of Development, Scalable and data storage etc .These parameters serve the actual purpose of comparing what new implementations are being carried out in our innovative project. Comparison between existing system and proposed system “FP Tree Using Spark" is shown in table 4.1.

Table 4.1 – Comparison between existing and proposed system.

Parameter	**Existing System**	**FP Tree Using Spark**
Scalability	Limitation in handling large dataset	Efficiently handle large-scale datasets
Performance	Performance issues, particularly as the data volume increases.	Expected to offer significantly better performance
Efficiency	It does not take full advantage of the available cluster resources.	More efficient in terms of resource utilization and execution time
Ease of Development	Challenging and Time Consuming	Provides high-level APIs that reduce development time
Scalable Data Storage	Limited support for scalable & distributed data storage options	Flexible and scalable data storage options.
Support Complex Workflows	Difficult to integrate with other data preprocessing	Combine them into a single process in a seamless manner.
Real Time Processing	There may be restrictions on real-time or streaming data processing	Can be integrated if real-time processing is required

Conclusion

For large-scale frequent itemset mining, Apache Spark implementation of the FP Tree algorithm offers a reliable option. Performance is significantly improved by this method, which makes use of Spark's distributed computing capabilities to handle large amounts of transaction data in an effective manner. Spark's distributed architecture makes the most of cluster resources, including CPU and memory, by ensuring optimal resource utilization. Additionally, the system is flexible, making it possible to integrate different data sources seamlessly, create intricate processes for data processing, and make development easier using high-level libraries and APIs. Apache Spark is flexible enough to accommodate a broad range of applications since it can be configured to enable batch and real-time data processing. Using FP Tree with Spark allows transaction data to yield insightful information that may be used for market basket analysis, recommendation engines, and other methods of making decisions based on data. Essentially, this approach is critical for big data-handling organizations because it makes it easier to identify patterns and links in large datasets, enabling them to optimize business processes and make well-informed decisions.

Appendix

1) https://app.creately.com/

Creately tremendously helped is making the UML diagrams in the project. The various UML diagrams made in the project are – Data Flow Diagrams, Use Case Diagrams and the Entity Relationship Diagrams.

2) PySpark

PySpark was used in this project for writing python code.

3) Data Samples

Provide sample transaction data or a data dictionary to help readers understand the structure and format of the data used in the implementation.

4) Code Snippets

Provide pertinent code snippets, particularly if your implementation requires the creation of unique functions or algorithms. This aids readers in understanding the reasoning and specifics of execution.

5) Configuration Detail

Share cluster configuration details like node count, hardware specifications, and Spark settings to enable others to replicate your setup.

6) Input & Output

The implementation results are explained through sample data, matching support values, and frequently occurring itemsets in the output.

References

[1] Liu, Y., Liao, W.K., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 689–695.Springer, Berlin (2018).

[2] Chen, Y., An, A.: Approximate parallel high utility itemset mining. Big Data Res. 6, 26–42 (2016)

[3] Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: SPMF: a Java open-source pattern mining library. J. Mach. Learn. Research 15(1), 3389–3393 (2014).

[4] Tseng, V.S., Wu, C.W., Fournier-Viger, P., Philip, S.Y.: Efficient algorithms for mining the concise and lossless representation of high utility itemsets. IEEE Trans. Knowl. Data Eng. 27(3), 726–739 (2014).

[5] Yao, H., Hamilton, H.J.: Mining itemset utilities from transaction databases. Data Knowl. Eng. 59(3), 603–626 (2006).

[6] Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases.

[7] Fournier-Viger, P., Zhang, Y., Lin, J.C.W., Fujita, H., Koh, Y.S.: Mining local and peak high utility itemsets. Inf. Sci. 481, 344–367 (2019)

[8] Sethi, K.K., Ramesh, D., Edla, D.R.: P-FHM+: Parallel high utility itemset mining algorithm for big data processing. Proceed. Comput. Sci. 132, 918–927 (2018)

[9] Matei Zaharia. An architecture for fast and general data processing on large clusters. Technical Report No. UCB/EECS-2014-12, University of California at Berkeley.

[10] Jeffrey Dean, Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In OSDI, 2004, p 137-150.

Acknowledgement

We take this opportunity to express our deep sense of gratitude to our project guide and project coordinator**, Mr. John Kenny** for his continuous guidance and encouragement throughout the duration of our major project work. Because of his experience and wonderful knowledge, we can fulfill the requirement of completing the major project within the stipulated time. We would also like to thank **Dr. Neeta Patil**, Head of Data Engineering department for his encouragement, whole-hearted cooperation and support.

We would also like to thank our Principal, **Dr. J. B. Patil** and the management of Universal College of Engineering, Vasai, Mumbai for providing us all the facilities and the work friendly environment. We acknowledge with thanks, the assistance provided by departmental staff, library and lab attendants.

Mr. Karan Singh (170)

Mr. Muhammad Sadiq (169)

Mr.Taher Sadriwala (163)

Mr. Parth Mistry (156)