NAME: Hem Ramesh Thumar

NJIT UCID: ht296

Email Address: ht296@njit.edu

13th October 2024

Professor: Yasser Abduallah

CS 634 - Data Mining

# Midterm Project Report

*Implementation and Code Usage*

---

## Apriori Algorithm Implementation in Retail Data Mining

## Abstract

This project explores the Apriori Algorithm, a fundamental data mining technique, to uncover associations within retail transactions. The algorithm is implemented, and various data mining concepts, principles, and methods are employed to assess its effectiveness and efficiency. Custom data mining tools are developed to create a model for extracting valuable insights from transaction data.

## Introduction

Data mining is a powerful approach for uncovering hidden patterns and associations within large datasets. This project focuses on the Apriori Algorithm, a classic method for association rule mining, and its application in a retail context. The core data mining concepts and principles applied in the work are outlined.

The main idea behind the Apriori Algorithm is to create associations. To do this, the algorithm first identifies the most frequent items in the list of transactions. Then, based on a user-defined support parameter, the support value for each item is calculated. Items that do not meet the user-defined support parameter are eliminated, and the remaining items are used to create associations. The Apriori algorithm is a well-known data mining technique that uses a brute-force approach to find frequently occurring sets of items and generate association rules. It works by repeatedly increasing the size of the item sets and removing those that don't meet a minimum support threshold.

In this particular implementation, the Apriori algorithm was applied to a custom dataset related to a retail store. This process involved several key steps:

1. Creating dictionaries to store candidate and frequent item sets.
2. Loading the dataset and item sets from CSV files.
3. Preprocessing the dataset to ensure proper item order and uniqueness.
4. Collecting user input for the minimum support and confidence thresholds.
5. Iteratively generating candidate item sets and updating the frequent item sets using the Apriori algorithm's brute-force approach, which considers all possible combinations of items.

This implementation allows us to identify the frequent item sets and association rules within the retail store's data, providing valuable insights for business decisions and strategies.

## Core concepts and Principles

### Frequent Itemset Discovery

The Apriori Algorithm is a fundamental method used in data mining that focuses on discovering frequent itemsets. These itemsets are essentially groups of items that frequently appear together in transactions. Understanding these itemsets is crucial as they provide valuable insights into customer purchasing behavior and preferences, helping businesses tailor their strategies to meet consumer needs more effectively.

### Support and Confidence

In the realm of data mining, two essential metrics play a pivotal role: support and confidence. Support is a measure that indicates how often a particular item or itemset appears in the dataset. It helps us understand the popularity of items among customers. On the other hand, confidence assesses the likelihood that items will be purchased together. This metric is vital for determining the strength of the relationship between items. Together, these metrics guide our analysis and help us make informed decisions based on the data.

### Association Rules

By identifying strong association rules, I can determine which items are commonly purchased together. These rules are not just theoretical; they are instrumental in optimizing sales strategies. For instance, they can be used to create effective product recommendations, enhancing the shopping experience for customers and potentially increasing sales for retailers. Understanding these associations allows businesses to strategically position products and create promotions that resonate with their target audience.

### Project Workflow

Our project follows a structured workflow that involves several stages, all centered around the application of the Apriori Algorithm:

### Data Loading and Preprocessing

The first step in our project is to load transaction data from a retail store dataset. Each transaction consists of a list of items that a customer has purchased. To ensure the accuracy and reliability of our analysis, we preprocess the dataset. This preprocessing step includes filtering out unique items and sorting them based on a predefined order. By doing this, we prepare the data for further analysis, ensuring that it is clean and organized.

**Determination of Minimum Support and Confidence**

User input is crucial in the data mining process. We actively collect the user's preferences regarding the minimum support and confidence levels they wish to set. This input is essential as it helps filter out less significant patterns that may not provide valuable insights. By establishing these thresholds, we can focus our analysis on the most relevant and impactful itemsets.

**Iteration Through Candidate Itemsets**

The iterative application of the Apriori Algorithm involves generating candidate itemsets of increasing sizes. We start with single items, referred to as itemset size K = 1, and then progress to K = 2, K = 3, and so forth. This iterative process employs a "brute force" method, where we generate all possible combinations of itemsets. This thorough approach ensures that we do not overlook any potential associations.

**Support Count Calculation**

For each candidate itemset generated, we calculate its support by counting how many transactions contain that specific itemset. Itemsets that meet or exceed the minimum support threshold are retained for further analysis, while those that do not meet this criterion are discarded. This step is crucial for narrowing down our focus to the most relevant itemsets.

**Confidence Calculation**

Next, we evaluate the confidence of the association rules. This step indicates the strength of the associations between items. It requires careful comparison of support values for individual items and itemsets. By analyzing these values, we can determine which associations are strong and worth considering in our recommendations.

**Association Rule Generation**

Finally, we extract association rules that satisfy both the minimum support and minimum confidence requirements. These rules reveal valuable insights into which items are often purchased together. By leveraging these insights, businesses can make data-driven decisions that enhance their marketing strategies and improve customer satisfaction.

## Result and Evaluation

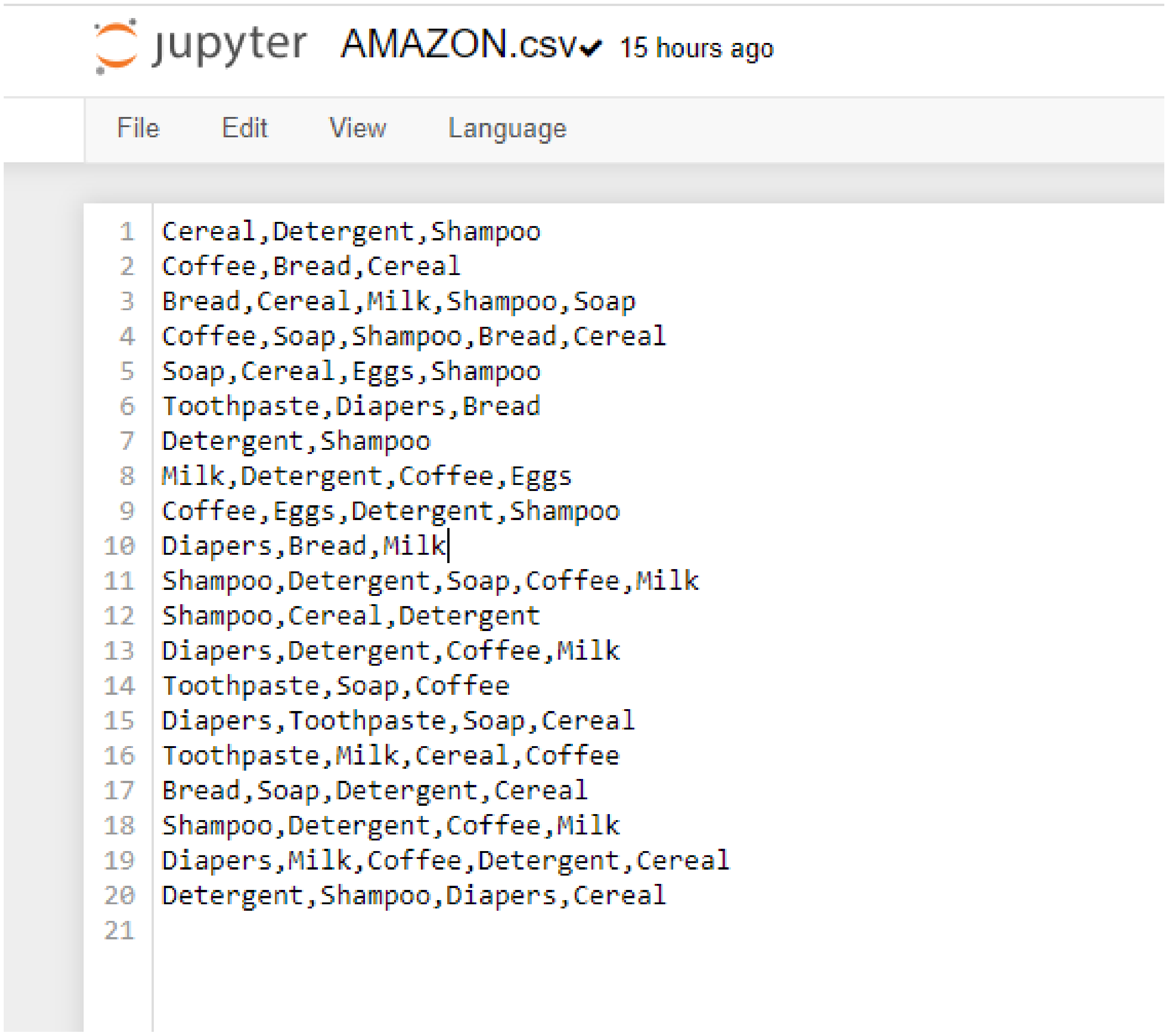
The project's effectiveness and efficiency are thoroughly evaluated using specific performance measures, which include support, confidence, and the resulting association rules derived from the data. These metrics help us understand how well the algorithm performs in identifying relationships within the data. Additionally, we conduct a comparison between our custom implementation of the Apriori Algorithm and the established Apriori library. This comparison is crucial as it allows us to assess the reliability and accuracy of our approach, ensuring that our results are both valid and trustworthy.

## Conclusion

In conclusion, our project effectively demonstrates the practical application of data mining concepts, principles, and methods in a real-world context. We successfully implemented the Apriori Algorithm to extract meaningful association rules from retail transaction data, showcasing its utility. The iterative, "brute force" approach, along with our custom algorithm design and strict adherence to user-defined parameters, exemplifies the power of data mining. This process reveals valuable patterns that can significantly enhance decision-making in the retail industry, ultimately leading to improved business strategies and outcomes.

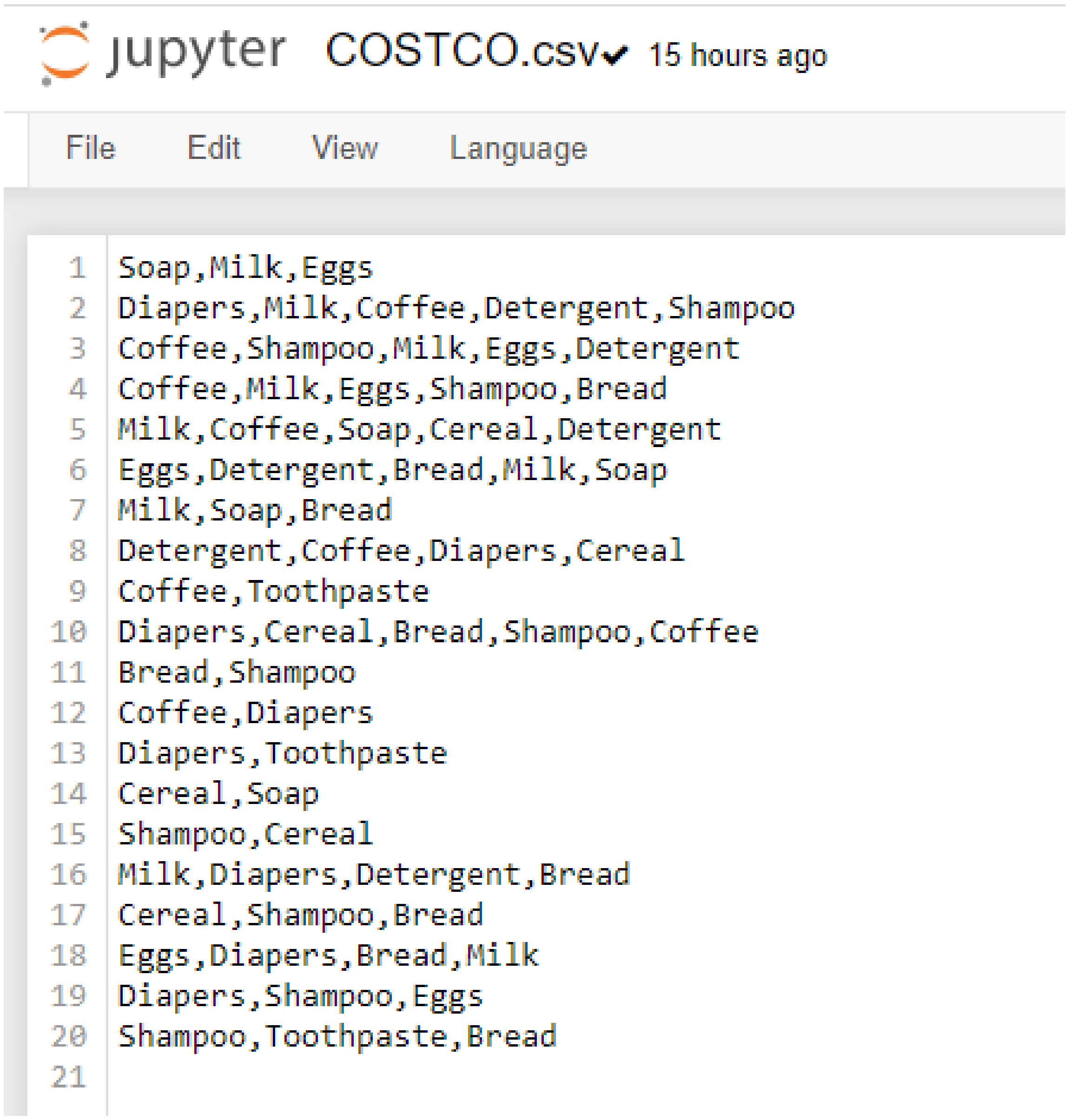
## Screenshots

Here are the csv files for the project.



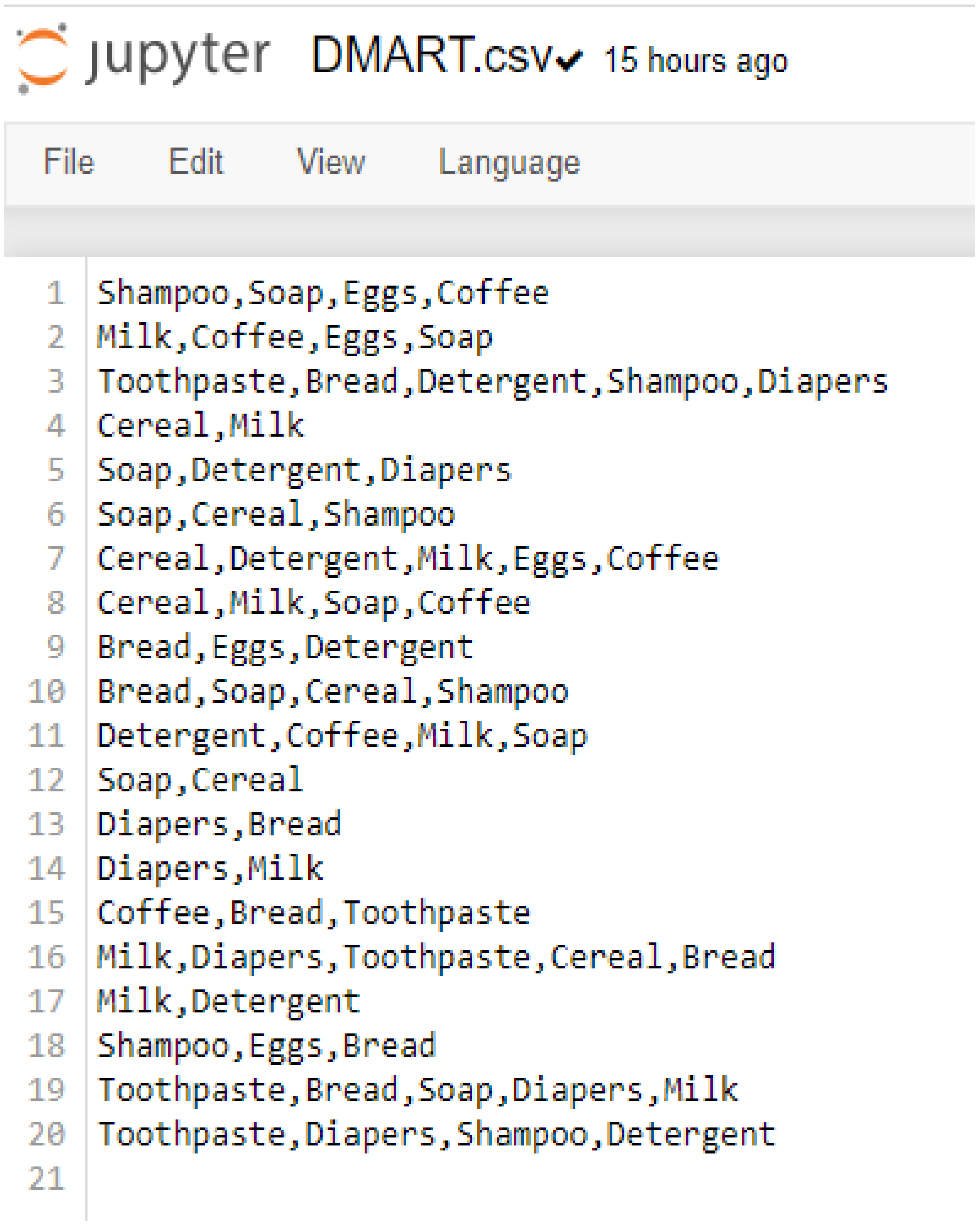
```
Cereal,Detergent,Shampoo
Coffee,Bread,Cereal
Bread,Cereal,Milk,Shampoo,Soap
Coffee,Soap,Shampoo,Bread,Cereal
Soap,Cereal,Eggs,Shampoo
Toothpaste,Diapers,Bread
Detergent,Shampoo
Milk,Detergent,Coffee,Eggs
Coffee,Eggs,Detergent,Shampoo
Diapers,Bread,Milk
Shampoo,Detergent,Soap,Coffee,Milk
Shampoo,Cereal,Detergent
Diapers,Detergent,Coffee,Milk
Toothpaste,Soap,Coffee
Diapers,Toothpaste,Soap,Cereal
Toothpaste,Milk,Cereal,Coffee
Bread,Soap,Detergent,Cereal
Shampoo,Detergent,Coffee,Milk
Diapers,Milk,Coffee,Detergent,Cereal
Detergent,Shampoo,Diapers,Cereal

```

*Fig1: Amazon_CSV*



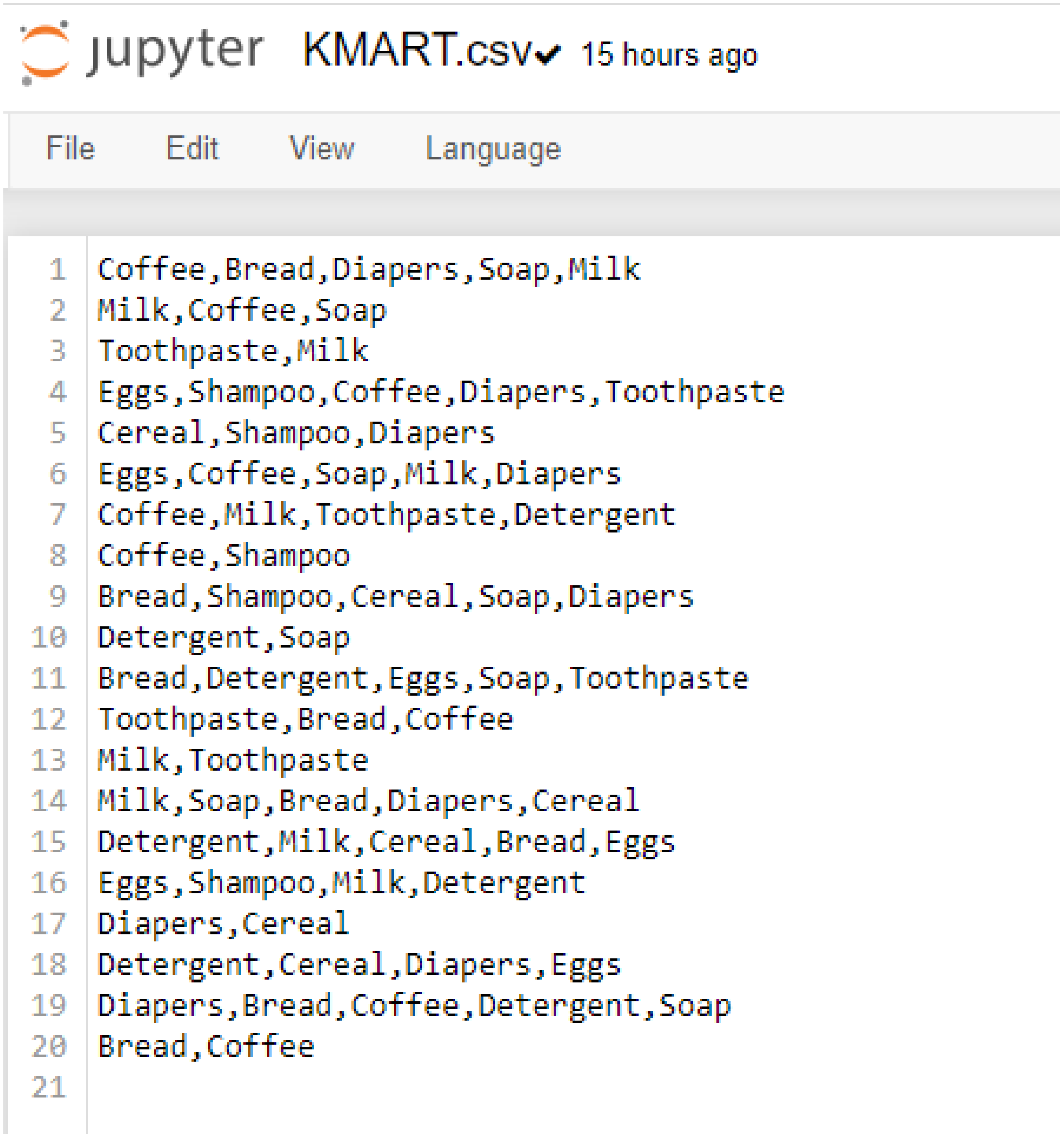
```
Soap,Milk,Eggs
Diapers,Milk,Coffee,Detergent,Shampoo
Coffee,Shampoo,Milk,Eggs,Detergent
Coffee,Milk,Eggs,Shampoo,Bread
Milk,Coffee,Soap,Cereal,Detergent
Eggs,Detergent,Bread,Milk,Soap
Milk,Soap,Bread
Detergent,Coffee,Diapers,Cereal
Coffee,Toothpaste
Diapers,Cereal,Bread,Shampoo,Coffee
Bread,Shampoo
Coffee,Diapers
Diapers,Toothpaste
Cereal,Soap
Shampoo,Cereal
Milk,Diapers,Detergent,Bread
Cereal,Shampoo,Bread
Eggs,Diapers,Bread,Milk
Diapers,Shampoo,Eggs
Shampoo,Toothpaste,Bread
```

*Fig 2: Costco_CSV*



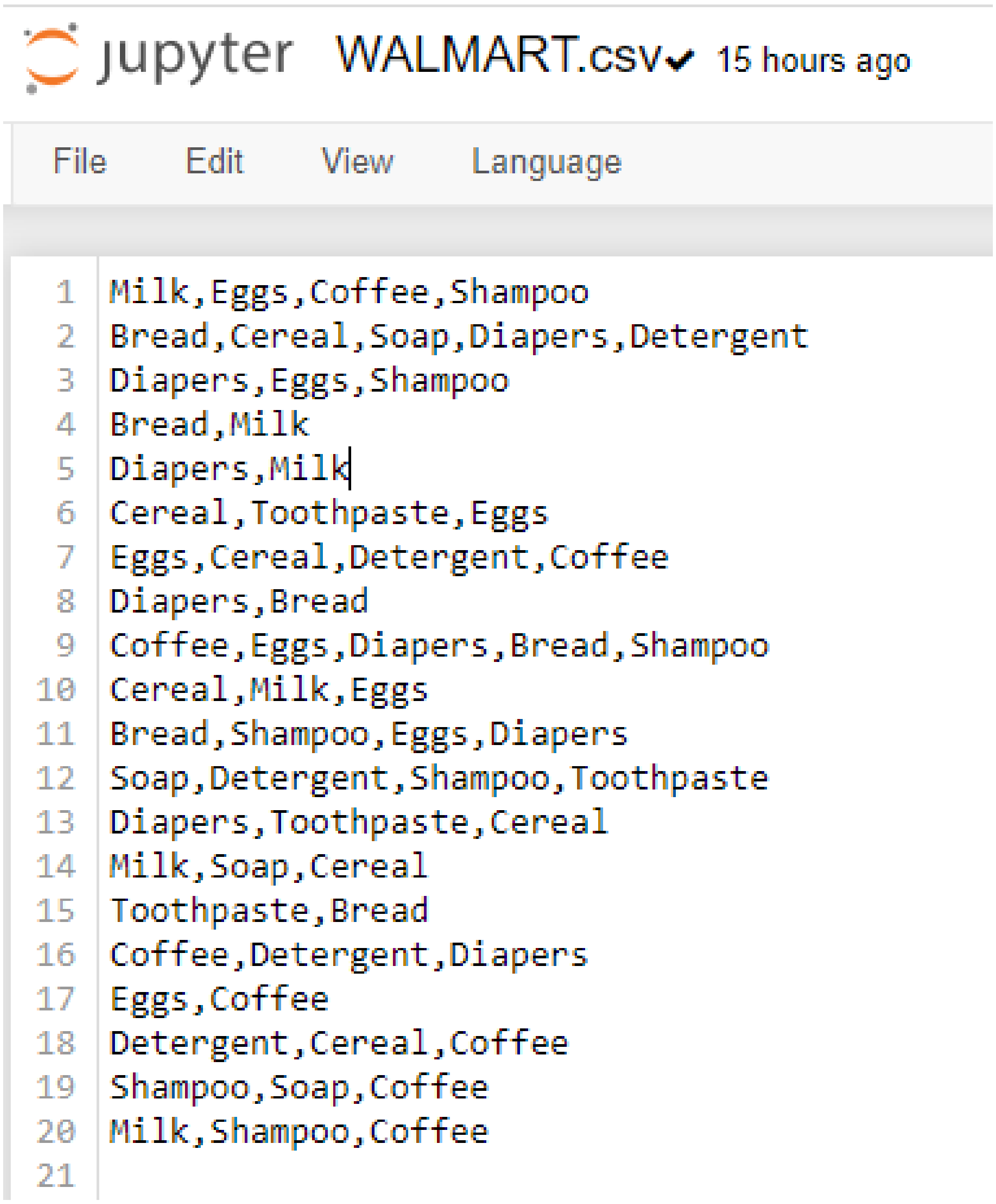
```
Shampoo,Soap,Eggs,Coffee
Milk,Coffee,Eggs,Soap
Toothpaste,Bread,Detergent,Shampoo,Diapers
Cereal,Milk
Soap,Detergent,Diapers
Soap,Cereal,Shampoo
Cereal,Detergent,Milk,Eggs,Coffee
Cereal,Milk,Soap,Coffee
Bread,Eggs,Detergent
Bread,Soap,Cereal,Shampoo
Detergent,Coffee,Milk,Soap
Soap,Cereal
Diapers,Bread
Diapers,Milk
Coffee,Bread,Toothpaste
Milk,Diapers,Toothpaste,Cereal,Bread
Milk,Detergent
Shampoo,Eggs,Bread
Toothpaste,Bread,Soap,Diapers,Milk
Toothpaste,Diapers,Shampoo,Detergent
```

*Fig 3: DMART_CSV*



```
Coffee,Bread,Diapers,Soap,Milk
Milk,Coffee,Soap
Toothpaste,Milk
Eggs,Shampoo,Coffee,Diapers,Toothpaste
Cereal,Shampoo,Diapers
Eggs,Coffee,Soap,Milk,Diapers
Coffee,Milk,Toothpaste,Detergent
Coffee,Shampoo
Bread,Shampoo,Cereal,Soap,Diapers
Detergent,Soap
Bread,Detergent,Eggs,Soap,Toothpaste
Toothpaste,Bread,Coffee
Milk,Toothpaste
Milk,Soap,Bread,Diapers,Cereal
Detergent,Milk,Cereal,Bread,Eggs
Eggs,Shampoo,Milk,Detergent
Diapers,Cereal
Detergent,Cereal,Diapers,Eggs
Diapers,Bread,Coffee,Detergent,Soap
Bread,Coffee
```

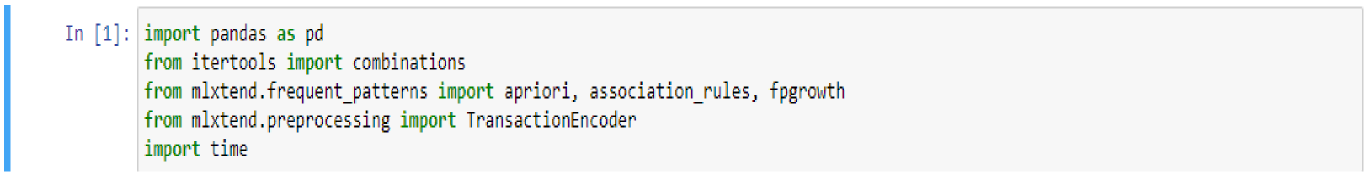
*Fig 4: KMART_CSV*



```
Milk,Eggs,Coffee,Shampoo
Bread,Cereal,Soap,Diapers,Detergent
Diapers,Eggs,Shampoo
Bread,Milk
Diapers,Milk
Cereal,Toothpaste,Eggs
Eggs,Cereal,Detergent,Coffee
Diapers,Bread
Coffee,Eggs,Diapers,Bread,Shampoo
Cereal,Milk,Eggs
Bread,Shampoo,Eggs,Diapers
Soap,Detergent,Shampoo,Toothpaste
Diapers,Toothpaste,Cereal
Milk,Soap,Cereal
Toothpaste,Bread
Coffee,Detergent,Diapers
Eggs,Coffee
Detergent,Cereal,Coffee
Shampoo,Soap,Coffee
Milk,Shampoo,Coffee

```

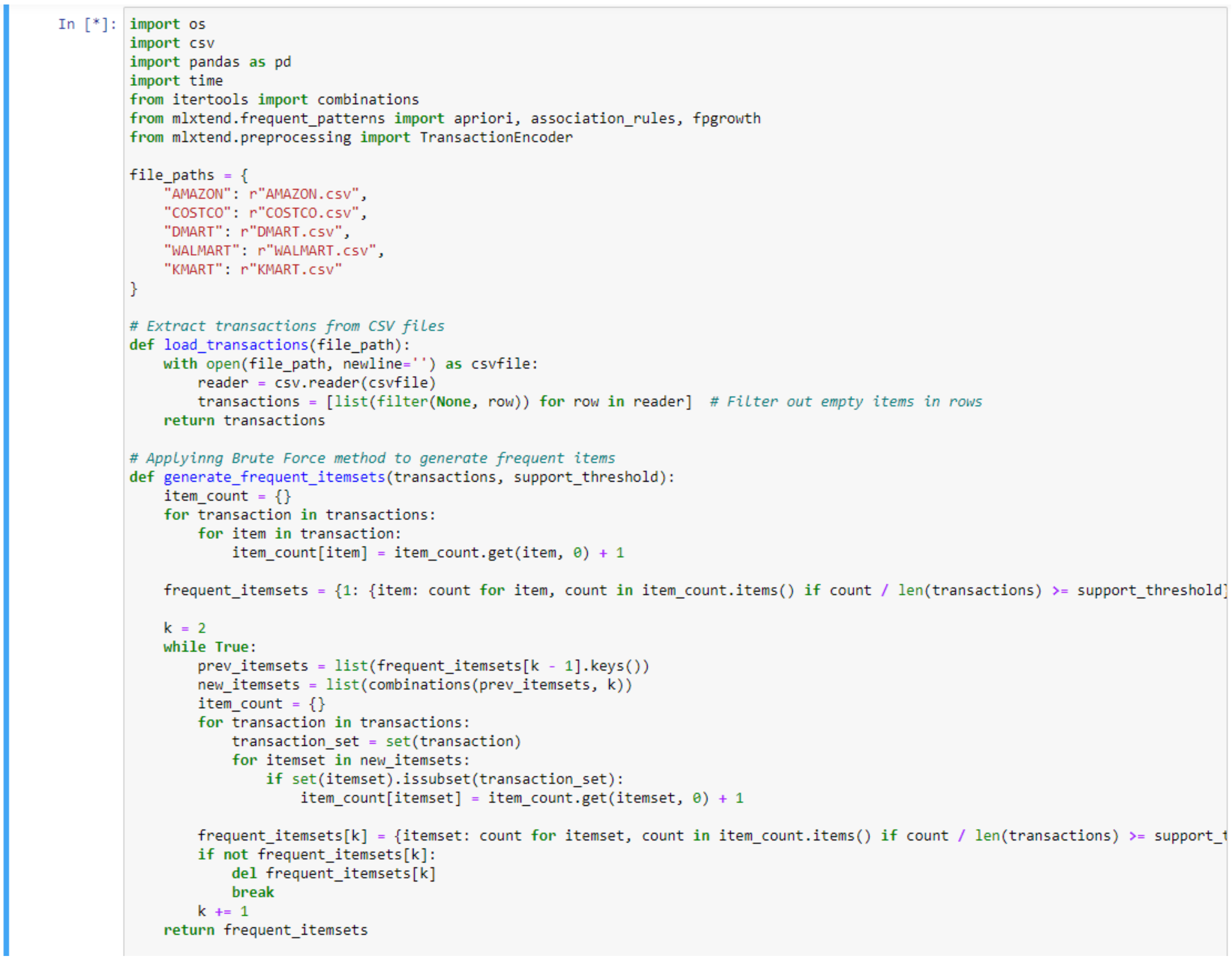
*Fig 5: Walmart_csv*

Below are the screenshots of the python code file:



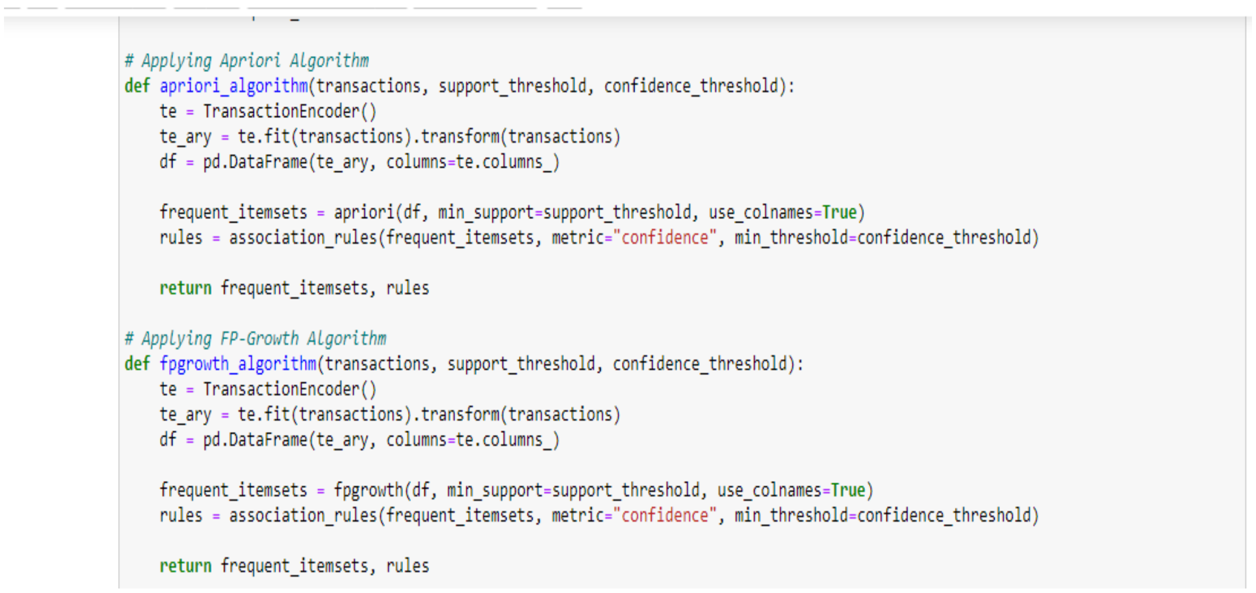
```
In [1]: import pandas as pd
        from itertools import combinations
        from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
        from mlxtend.preprocessing import TransactionEncoder
        import time
```

*Fig 1: Libraries used*



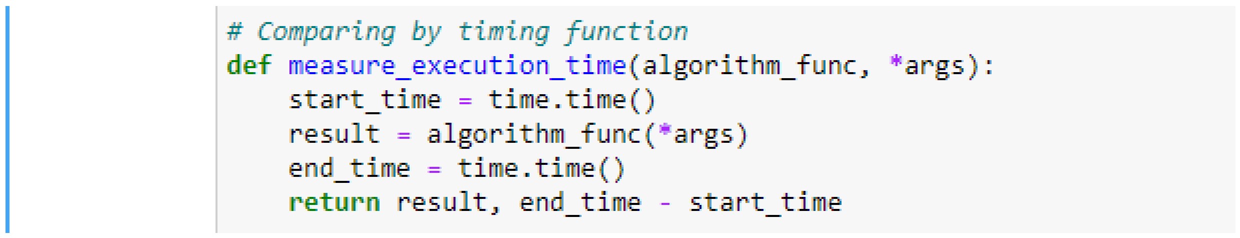
```
In [*]: import os
        import csv
        import pandas as pd
        import time
        from itertools import combinations
        from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
        from mlxtend.preprocessing import TransactionEncoder

        file_paths = {
            "AMAZON": r"AMAZON.csv",
            "COSTCO": r"COSTCO.csv",
            "DMART": r"DMART.csv",
            "WALMART": r"WALMART.csv",
            "KMART": r"KMART.csv"
        }

        # Extract transactions from CSV files
        def load_transactions(file_path):
            with open(file_path, newline='') as csvfile:
                reader = csv.reader(csvfile)
                transactions = [list(filter(None, row)) for row in reader]  # Filter out empty items in rows
            return transactions

        # Applyinng Brute Force method to generate frequent items
        def generate_frequent_itemsets(transactions, support_threshold):
            item_count = {}
            for transaction in transactions:
                for item in transaction:
                    item_count[item] = item_count.get(item, 0) + 1

            frequent_itemsets = {1: {item: count for item, count in item_count.items() if count / len(transactions) >= support_threshold}

            k = 2
            while True:
                prev_itemsets = list(frequent_itemsets[k - 1].keys())
                new_itemsets = list(combinations(prev_itemsets, k))
                item_count = {}
                for transaction in transactions:
                    transaction_set = set(transaction)
                    for itemset in new_itemsets:
                        if set(itemset).issubset(transaction_set):
                            item_count[itemset] = item_count.get(itemset, 0) + 1

                frequent_itemsets[k] = {itemset: count for itemset, count in item_count.items() if count / len(transactions) >= support_t
                if not frequent_itemsets[k]:
                    del frequent_itemsets[k]
                    break
                k += 1
            return frequent_itemsets
```

*Fig 2:Reading csv files and generating frequent items*



```
# Applying Apriori Algorithm
def apriori_algorithm(transactions, support_threshold, confidence_threshold):
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)

    frequent_itemsets = apriori(df, min_support=support_threshold, use_colnames=True)
    rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=confidence_threshold)

    return frequent_itemsets, rules

# Applying FP-Growth Algorithm
def fpgrowth_algorithm(transactions, support_threshold, confidence_threshold):
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)

    frequent_itemsets = fpgrowth(df, min_support=support_threshold, use_colnames=True)
    rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=confidence_threshold)

    return frequent_itemsets, rules
```

*Fig 3: Applying apriori and fp growth algorithm*



```
# Comparing by timing function
def measure_execution_time(algorithm_func, *args):
    start_time = time.time()
    result = algorithm_func(*args)
    end_time = time.time()
    return result, end_time - start_time
```

*Fig 4: Comparing time function*

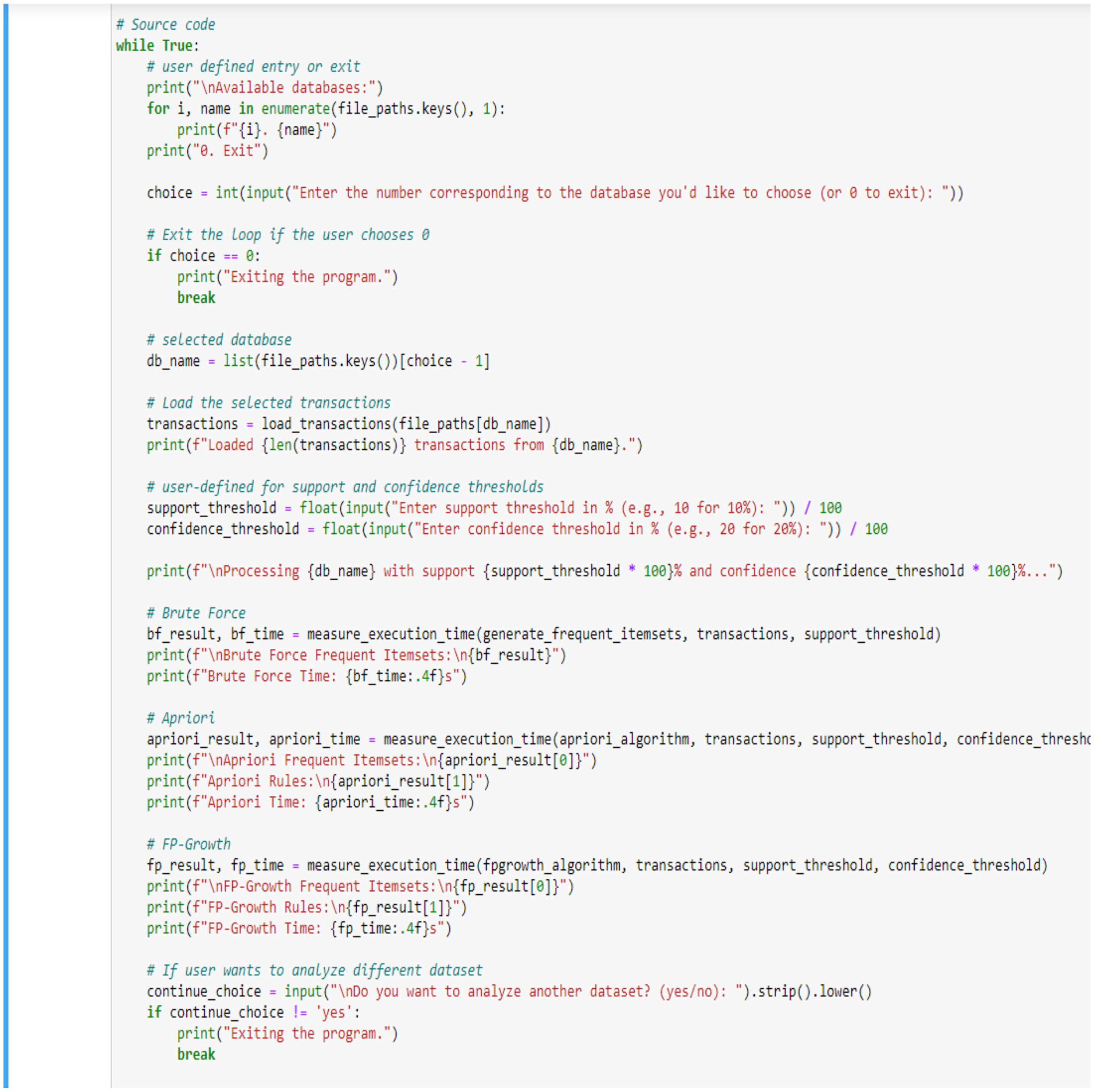
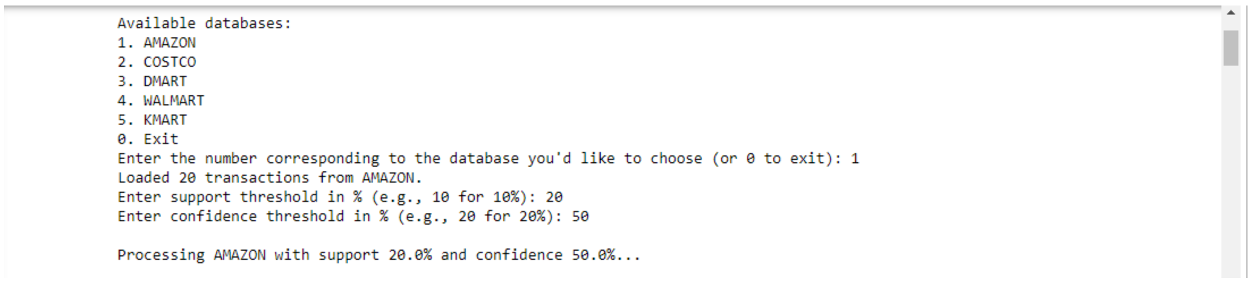

```
# Source code
while True:
    # user defined entry or exit
    print("\nAvailable databases:")
    for i, name in enumerate(file_paths.keys(), 1):
        print(f"{i}. {name}")
    print("0. Exit")

    choice = int(input("Enter the number corresponding to the database you'd like to choose (or 0 to exit): "))

    # Exit the loop if the user chooses 0
    if choice == 0:
        print("Exiting the program.")
        break

    # selected database
    db_name = list(file_paths.keys())[choice - 1]

    # Load the selected transactions
    transactions = load_transactions(file_paths[db_name])
    print(f"Loaded {len(transactions)} transactions from {db_name}.")

    # user-defined for support and confidence thresholds
    support_threshold = float(input("Enter support threshold in % (e.g., 10 for 10%): ")) / 100
    confidence_threshold = float(input("Enter confidence threshold in % (e.g., 20 for 20%): ")) / 100

    print(f"\nProcessing {db_name} with support {support_threshold * 100}% and confidence {confidence_threshold * 100}%...")

    # Brute Force
    bf_result, bf_time = measure_execution_time(generate_frequent_itemsets, transactions, support_threshold)
    print(f"\nBrute Force Frequent Itemsets:\n{bf_result}")
    print(f"Brute Force Time: {bf_time:.4f}s")

    # Apriori
    apriori_result, apriori_time = measure_execution_time(apriori_algorithm, transactions, support_threshold, confidence_thresho
    print(f"\nApriori Frequent Itemsets:\n{apriori_result[0]}")
    print(f"Apriori Rules:\n{apriori_result[1]}")
    print(f"Apriori Time: {apriori_time:.4f}s")

    # FP-Growth
    fp_result, fp_time = measure_execution_time(fpgrowth_algorithm, transactions, support_threshold, confidence_threshold)
    print(f"\nFP-Growth Frequent Itemsets:\n{fp_result[0]}")
    print(f"FP-Growth Rules:\n{fp_result[1]}")
    print(f"FP-Growth Time: {fp_time:.4f}s")

    # If user wants to analyze different dataset
    continue_choice = input("\nDo you want to analyze another dataset? (yes/no): ").strip().lower()
    if continue_choice != 'yes':
        print("Exiting the program.")
        break
```

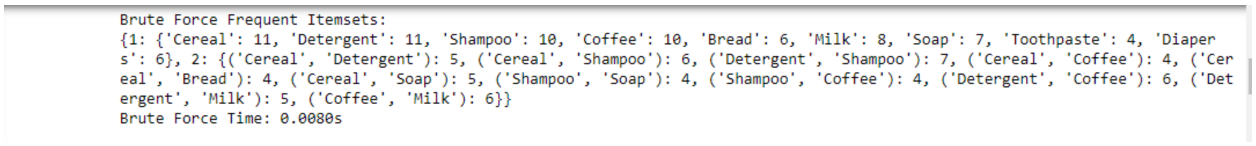
*Fig 5: Main source code*

## Output:



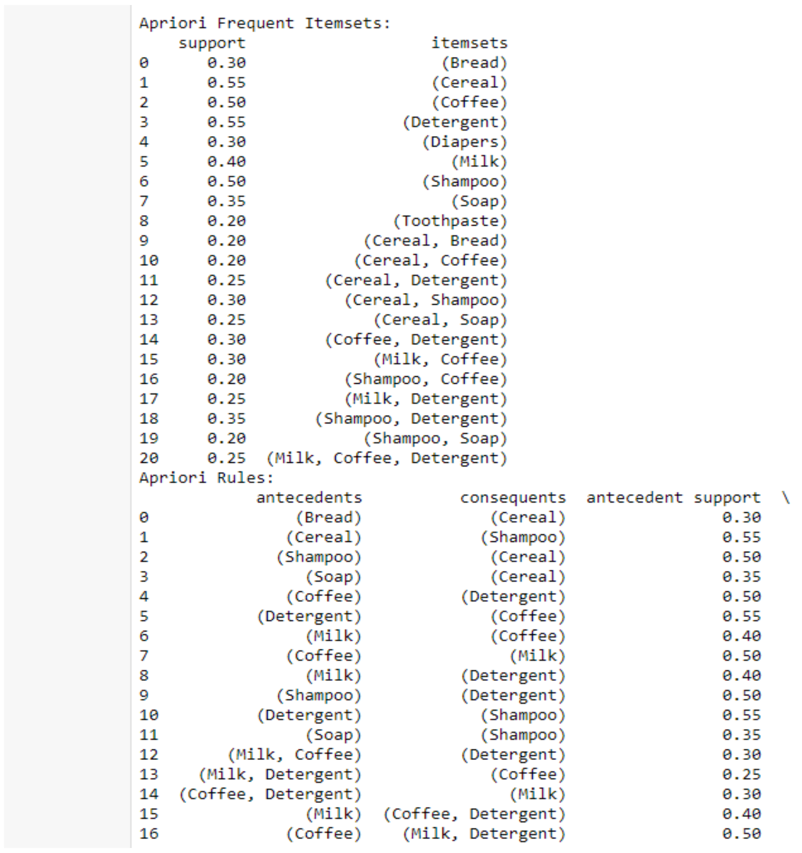
```
Available databases:
1. AMAZON
2. COSTCO
3. DMART
4. WALMART
5. KMART
0. Exit
Enter the number corresponding to the database you'd like to choose (or 0 to exit): 1
Loaded 20 transactions from AMAZON.
Enter support threshold in % (e.g., 10 for 10%): 20
Enter confidence threshold in % (e.g., 20 for 20%): 50

Processing AMAZON with support 20.0% and confidence 50.0%...
```

*Fig 1: User defined datasets*



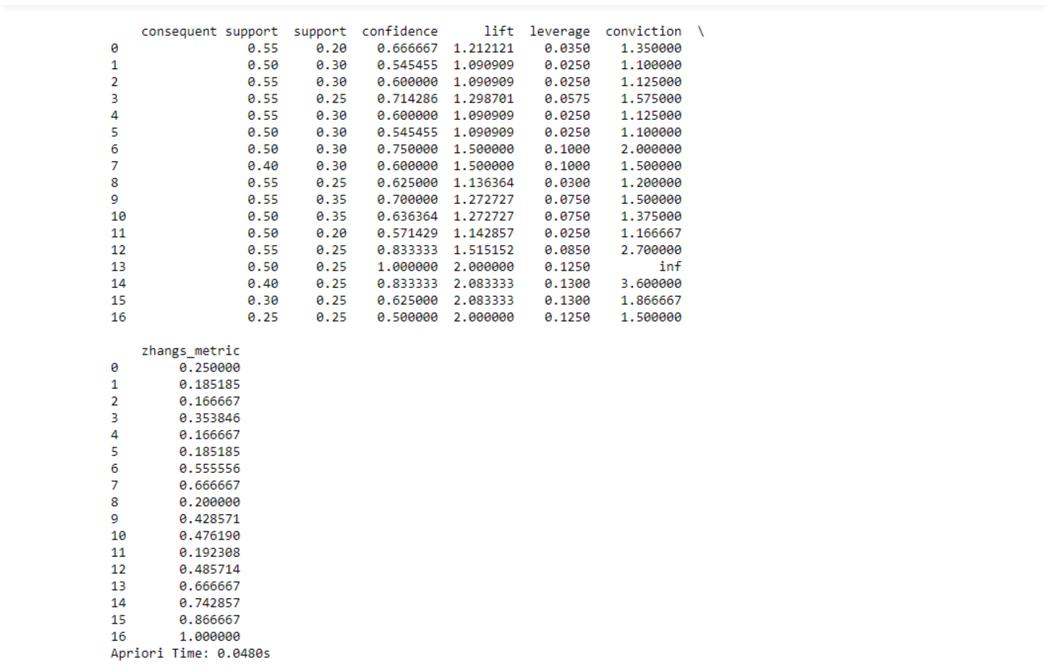
```
Brute Force Frequent Itemsets:
{1: {'Cereal': 11, 'Detergent': 11, 'Shampoo': 10, 'Coffee': 10, 'Bread': 6, 'Milk': 8, 'Soap': 7, 'Toothpaste': 4, 'Diaper
s': 6}, 2: {('Cereal', 'Detergent'): 5, ('Cereal', 'Shampoo'): 6, ('Detergent', 'Shampoo'): 7, ('Cereal', 'Coffee'): 4, ('Cer
eal', 'Bread'): 4, ('Cereal', 'Soap'): 5, ('Shampoo', 'Soap'): 4, ('Shampoo', 'Coffee'): 4, ('Detergent', 'Coffee'): 6, ('Det
ergent', 'Milk'): 5, ('Coffee', 'Milk'): 6}}
Brute Force Time: 0.0080s
```

*Fig 2: Brute Force time*



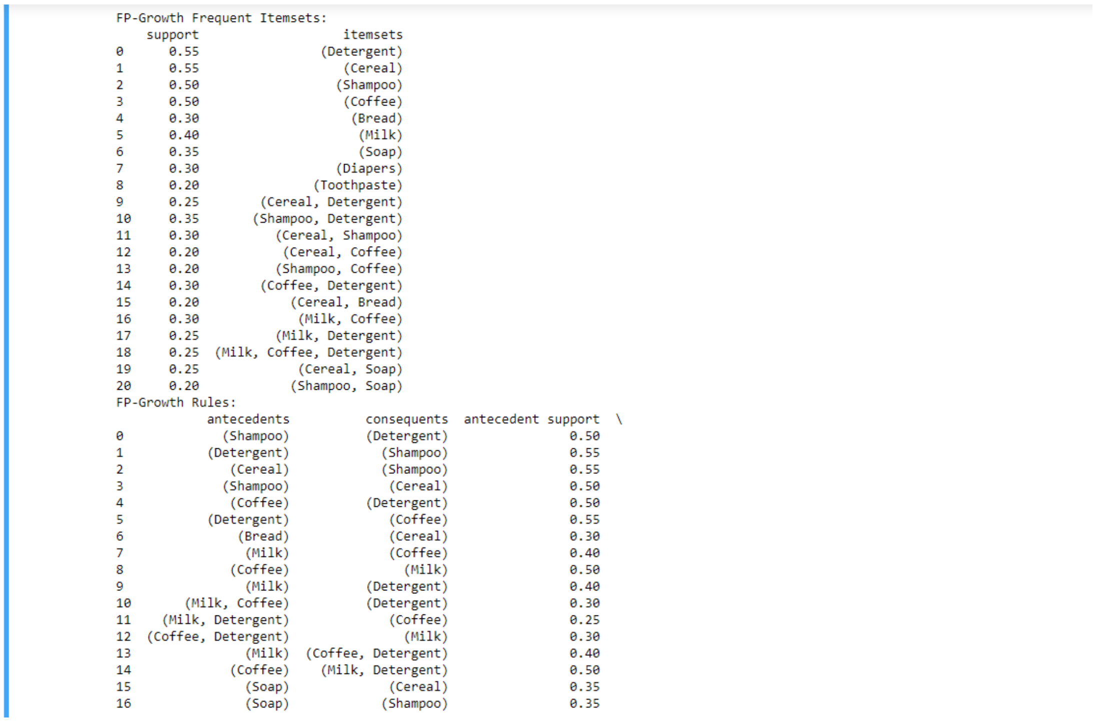
```
Apriori Frequent Itemsets:
    support                  itemsets
0      0.30                   (Bread)
1      0.55                  (Cereal)
2      0.50                  (Coffee)
3      0.55               (Detergent)
4      0.30                 (Diapers)
5      0.40                    (Milk)
6      0.50                 (Shampoo)
7      0.35                    (Soap)
8      0.20              (Toothpaste)
9      0.20           (Cereal, Bread)
10     0.20          (Cereal, Coffee)
11     0.25       (Cereal, Detergent)
12     0.30         (Cereal, Shampoo)
13     0.25            (Cereal, Soap)
14     0.30       (Coffee, Detergent)
15     0.30            (Milk, Coffee)
16     0.20         (Shampoo, Coffee)
17     0.25         (Milk, Detergent)
18     0.35      (Shampoo, Detergent)
19     0.20           (Shampoo, Soap)
20     0.25  (Milk, Coffee, Detergent)
Apriori Rules:
            antecedents          consequents  antecedent support  \
0               (Bread)             (Cereal)                0.30
1              (Cereal)            (Shampoo)                0.55
2             (Shampoo)             (Cereal)                0.50
3                (Soap)             (Cereal)                0.35
4              (Coffee)          (Detergent)                0.50
5           (Detergent)             (Coffee)                0.55
6                (Milk)             (Coffee)                0.40
7              (Coffee)               (Milk)                0.50
8                (Milk)          (Detergent)                0.40
9             (Shampoo)          (Detergent)                0.50
10          (Detergent)            (Shampoo)                0.55
11               (Soap)            (Shampoo)                0.35
12       (Milk, Coffee)          (Detergent)                0.30
13    (Milk, Detergent)             (Coffee)                0.25
14  (Coffee, Detergent)               (Milk)                0.30
15               (Milk)  (Coffee, Detergent)                0.40
16             (Coffee)    (Milk, Detergent)                0.50
```

*Fig 3: Apriori algorithm and rules*



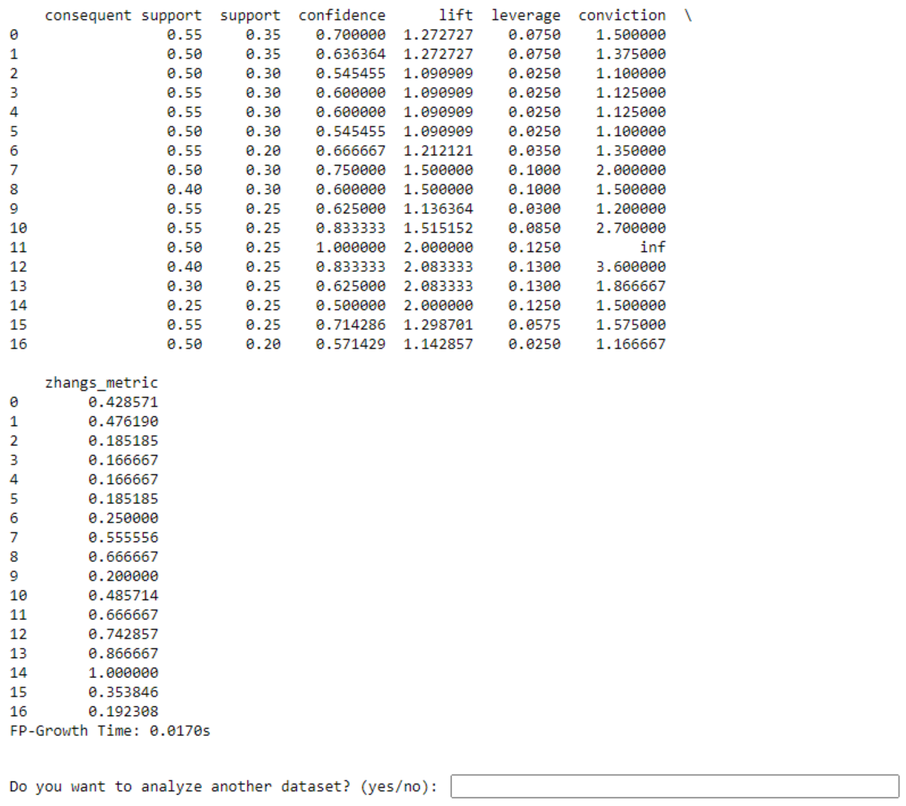
```
    consequent support  support  confidence      lift  leverage  conviction  \
0                 0.55     0.20    0.666667  1.212121    0.0350    1.350000
1                 0.50     0.30    0.545455  1.090909    0.0250    1.100000
2                 0.55     0.30    0.600000  1.090909    0.0250    1.125000
3                 0.55     0.25    0.714286  1.298701    0.0575    1.575000
4                 0.55     0.30    0.600000  1.090909    0.0250    1.125000
5                 0.50     0.30    0.545455  1.090909    0.0250    1.100000
6                 0.50     0.30    0.750000  1.500000    0.1000    2.000000
7                 0.40     0.30    0.600000  1.500000    0.1000    1.500000
8                 0.55     0.25    0.625000  1.136364    0.0300    1.200000
9                 0.55     0.35    0.700000  1.272727    0.0750    1.500000
10                0.50     0.35    0.636364  1.272727    0.0750    1.375000
11                0.50     0.20    0.571429  1.142857    0.0250    1.166667
12                0.55     0.25    0.833333  1.515152    0.0850    2.700000
13                0.50     0.25    1.000000  2.000000    0.1250         inf
14                0.40     0.25    0.833333  2.083333    0.1300    3.600000
15                0.30     0.25    0.625000  2.083333    0.1300    1.866667
16                0.25     0.25    0.500000  2.000000    0.1250    1.500000

    zhangs_metric
0        0.250000
1        0.185185
2        0.166667
3        0.353846
4        0.166667
5        0.185185
6        0.555556
7        0.666667
8        0.200000
9        0.428571
10       0.476190
11       0.192308
12       0.485714
13       0.666667
14       0.742857
15       0.866667
16       1.000000
Apriori Time: 0.0480s
```

*Fig 4: Calculations and metrics*



```
FP-Growth Frequent Itemsets:
    support                   itemsets
0      0.55                (Detergent)
1      0.55                   (Cereal)
2      0.50                  (Shampoo)
3      0.50                   (Coffee)
4      0.30                    (Bread)
5      0.40                     (Milk)
6      0.35                     (Soap)
7      0.30                  (Diapers)
8      0.20                (Toothpaste)
9      0.25          (Cereal, Detergent)
10     0.35         (Shampoo, Detergent)
11     0.30            (Cereal, Shampoo)
12     0.20             (Cereal, Coffee)
13     0.20            (Shampoo, Coffee)
14     0.30          (Coffee, Detergent)
15     0.20              (Cereal, Bread)
16     0.30               (Milk, Coffee)
17     0.25            (Milk, Detergent)
18     0.25    (Milk, Coffee, Detergent)
19     0.25               (Cereal, Soap)
20     0.20              (Shampoo, Soap)
FP-Growth Rules:
            antecedents          consequents  antecedent support  \
0             (Shampoo)          (Detergent)                0.50
1           (Detergent)            (Shampoo)                0.55
2              (Cereal)            (Shampoo)                0.55
3             (Shampoo)             (Cereal)                0.50
4              (Coffee)          (Detergent)                0.50
5           (Detergent)             (Coffee)                0.55
6               (Bread)             (Cereal)                0.30
7                (Milk)             (Coffee)                0.40
8              (Coffee)               (Milk)                0.50
9                (Milk)          (Detergent)                0.40
10       (Milk, Coffee)          (Detergent)                0.30
11    (Milk, Detergent)             (Coffee)                0.25
12  (Coffee, Detergent)               (Milk)                0.30
13               (Milk)  (Coffee, Detergent)                0.40
14             (Coffee)    (Milk, Detergent)                0.50
15               (Soap)             (Cereal)                0.35
16               (Soap)            (Shampoo)                0.35
```

*Fig 5: FP-Growth algorithm and rules*



```
    consequent support  support  confidence      lift  leverage  conviction  \
0                 0.55     0.35    0.700000  1.272727    0.0750    1.500000
1                 0.50     0.35    0.636364  1.272727    0.0750    1.375000
2                 0.50     0.30    0.545455  1.090909    0.0250    1.100000
3                 0.55     0.30    0.600000  1.090909    0.0250    1.125000
4                 0.55     0.30    0.600000  1.090909    0.0250    1.125000
5                 0.50     0.30    0.545455  1.090909    0.0250    1.100000
6                 0.55     0.20    0.666667  1.212121    0.0350    1.350000
7                 0.50     0.30    0.750000  1.500000    0.1000    2.000000
8                 0.40     0.30    0.600000  1.500000    0.1000    1.500000
9                 0.55     0.25    0.625000  1.136364    0.0300    1.200000
10                0.55     0.25    0.833333  1.515152    0.0850    2.700000
11                0.50     0.25    1.000000  2.000000    0.1250         inf
12                0.40     0.25    0.833333  2.083333    0.1300    3.600000
13                0.30     0.25    0.625000  2.083333    0.1300    1.866667
14                0.25     0.25    0.500000  2.000000    0.1250    1.500000
15                0.55     0.25    0.714286  1.298701    0.0575    1.575000
16                0.50     0.20    0.571429  1.142857    0.0250    1.166667

    zhangs_metric
0        0.428571
1        0.476190
2        0.185185
3        0.166667
4        0.166667
5        0.185185
6        0.250000
7        0.555556
8        0.666667
9        0.200000
10       0.485714
11       0.666667
12       0.742857
13       0.866667
14       1.000000
15       0.353846
16       0.192308
FP-Growth Time: 0.0170s

Do you want to analyze another dataset? (yes/no):
```

*Fig 6: calculations and metrics for fp-growth*

## Other

The source code (.py file) and data sets (.csv files) will be attached to the zip file.

Link to Git Repository –

https://github.com/HemThumar/DM_midtermproj