New Jersey Institute of Technology

Mid-Term Project Report

Apriori and FP Growth Algorithms Implementation Using Python

NAME: Bhumireddy Sai Teja Reddy
NJIT UCID: sb2744
Email Address: sb2744@njit.edu
Professor: Yasser Abduallah
CS 634 - 104 Data Mining

Contents

Abstract: 4
1. Introduction 4
2. Foundational Concepts 4
   Frequent Itemsets 4
   Association Rules 4
   Support: 4
   Confidence: 4
   Lift: 4
3. The Apriori Algorithm 4
   Implementation: 4
   Advantages 4
   Limitations: 5
   Real-World Applications: 5
4. The FP-Growth Algorithm 5
   Implementation: 5
   Advantages 5
   Limitations: 5
   Real-World Applications: 5
5. Comparison and Considerations 5
6. Detailed Steps of Implementation for Apriori 5
7.Detailed Steps of Implementation for FP Growth 6
Project Workflow: 7
1. Introduction 7
2. Dataset Selection 7
   1) Amazon 7
   2) Best Buy 8
   3) K-mart 9
   4) Nike 10
   5) Generic 10
3. User Inputs for Minimum Support and Confidence 12
4. Brute-Forced Apriori Implementation 12
5. Validation with Python Package of Apriori 15
6. Validation with FP-Growth Algorithm 17
7. Comparing time taken by each algorithm 18
8. Results based on different minimum support and confidence on each data set. 19

For Amazon Data set: 19
For BestBuy Dataset: 21
For Kmart Dataset: 24
For Nike Dataset: 26
For Generic Dataset: 27
9. Performance Analysis 29
Results 29
Conclusion: 29
Recommendations 29
Steps to run the Program 30
STEP 1: Clone the repository 30
STEP 2: Create Virtual Environment 30
STEP 3: Install required libraries 30
STEP 4: Run the python file. 30
Conclusion 31
References and Links: 31

Abstract:

This project conducts a thorough examination of transaction data extracted from an online retail dataset to unveil inherent patterns and correlations among customer purchases, leveraging the Apriori and FP-Growth algorithms. The objective is to unearth frequent item sets and establish association rules that shed light on customer buying behaviors, particularly focusing on programming and web development books. The study delves into the theoretical foundations of association rule learning and elucidates the deployment of both brute-force and optimized algorithmic methods to effectively analyze the dataset. Through this analytical endeavor, the project showcases the pragmatic utility of data mining in refining retail strategies, spanning inventory management to personalized marketing initiatives. The insights gleaned from this analysis aim to empower decision-makers within the retail sector to align their product offerings more adeptly with evolving customer preferences and market trends.

1. Introduction

Within the realm of data mining, various techniques and methodologies are employed to reveal patterns, associations, and correlations within extensive datasets. Notably, the Apriori and FP-Growth algorithms are distinguished for their effectiveness in mining frequent itemsets and facilitating association rule learning. These algorithms play a crucial role in deciphering consumer behavior, refining product recommendations, and streamlining inventory management processes.

2. Foundational Concepts

Prior to exploring the algorithms, it is essential to grasp the foundational principles that underlie data mining endeavors.

Frequent Itemsets: Collections of items that co-occur in a dataset with a frequency exceeding a threshold specified by the user.

Association Rules: Associations indicating the probability of itemsets occurring together, expressed as X⇒Y, where X and Y represent separate itemsets.

Support: The percentage of transactions within the dataset containing a particular itemset, denoting its occurrence rate.

Confidence: An indication of the frequency with which items in set Y occur in transactions containing set X, representing the credibility of the association.

Lift: This metric measures the potency of a rule relative to the random co-occurrence of X and Y, with values surpassing 1 indicating a positive association.

3. The Apriori Algorithm

Utilizing an iterative approach, the Apriori algorithm discerns frequent itemsets and association rules by leveraging the downward closure property, thereby optimizing the search space efficiently.**Implementation:** Iteratively generates candidate itemsets and eliminates those with support below the threshold.

Advantages: Simplicity and ease of implementation.

Limitations: Performance issues with large datasets due to multiple database scans and high computational overhead.

Real-World Applications: Market basket analysis, cross-marketing strategies, and catalog design.

4. The FP-Growth Algorithm

FP-Growth streamlines frequent itemset discovery without candidate generation, using a compact FP-tree structure to encapsulate data relationships.

Implementation: Constructs an FP-tree from the dataset, then mines frequent itemsets directly from the tree.

Advantages: Significantly faster than Apriori, especially on large datasets, and requires only two passes over the dataset.

Limitations: Complexity in understanding and implementing the FP-tree structure and potential memory constraints.

Real-World Applications: E-commerce recommendations, fraud detection in banking, and patient data analysis in healthcare.

5. Comparison and Considerations

Although Apriori provides a fundamental method for uncovering item associations, its scalability limitations render FP-Growth more suitable for managing extensive and intricate datasets. The selection of algorithms hinges on the particular needs of the analysis, considering factors such as dataset magnitude, itemset intricacy, and available computational capabilities.

6. Detailed Steps of Implementation for Apriori

1. **Collect Data**: Assemble a dataset of transactions where each transaction is a set of items purchased together.
2. **Set Minimum Support and Confidence**:
    - **Support** is the percentage of transactions that include a particular itemset.
    - **Confidence** is a measure of the likelihood that an itemset is purchased when another itemset is purchased.
3. **Generate Candidate Itemsets**:
    - Start with candidate itemsets of size 1 (C1), which are just the individual items.
4. **Determine Frequent Itemsets**:
    - Count the frequency of each candidate itemset in the dataset.
    - Compare it with the minimum support threshold.
    - Retain those that meet or exceed the threshold. These are your frequent itemsets (F1).
5. **Create Larger Itemsets**:

- Use the frequent itemsets found in the previous step to generate new candidate itemsets of larger sizes (C2, C3, etc.).
- This process involves joining itemsets with themselves and pruning subsets that are not frequent.

6. **Repeat Determination of Frequent Itemsets**:
   - Repeat steps 4 and 5 for larger and larger itemsets until no new frequent itemsets are found.
7. **Generate Association Rules**:
   - For every frequent itemset, generate all possible rules.
   - Calculate the confidence of each rule.
   - Keep only those rules that meet the minimum confidence threshold.

7.Detailed Steps of Implementation for FP Growth

1. **Collect Data**: Begin with a dataset of transactions, where each transaction is a list of items purchased together.
2. **Create the FP-Tree**:
   - **Step 1: Count item frequencies** and order items in individual transactions by descending frequency.
   - **Step 2: Build the tree**. Starting with a null root, add paths for transactions. Share nodes, when possible, which means that if two transactions have a common prefix, they share the initial portion of their paths.
3. **Mine Frequent Itemsets**:
   - For each item, starting from the least frequent, create a conditional base (a sub-dataset of transactions containing that item).
   - Build a conditional FP-Tree for this base.
   - Recursively mine this tree, appending the item to the generated suffixes, to find all frequent itemsets involving this item.
4. **Generate Association Rules**:
   - From the frequent itemsets discovered, generate rules, applying a minimum confidence threshold to filter these rules.

Project Workflow:

1. Introduction

This report provides a comprehensive overview of the process and findings from the analysis of transaction data using association rule learning algorithms. The primary objective is to utilize Apriori and FP-Growth algorithms to uncover meaningful patterns and associations among the transactions within various datasets including Amazon, BestBuy, K-Mart, Nike, and Generic.

2. Dataset Selection

The analysis begins with the selection of a dataset by the user. The available options are 1) Amazon, 2) BestBuy, 3) K-Mart, 4) Nike, and 5) Generic. Based on the user's choice, the corresponding dataset is loaded for analysis. This allows for flexibility in the analysis and the opportunity to explore different transaction patterns across various retail contexts.

Below are the datasets used in this project. You can give any of the dataset to the program unless the format is same as below.

1) Amazon

Item #	Item Name
1	A Beginner’s Guide
2	Java: The Complete Reference
3	Java For Dummies
4	Android Programming: The Big Nerd Ranch
5	Head First Java 2nd Edition
6	Beginning Programming with Java
7	Java 8 Pocket Guide
8	C++ Programming in Easy Steps
9	Effective Java (2nd Edition)
10	HTML and CSS: Design and Build Websites

Table 1 Amazon Item Names

Transaction ID	Transaction
Trans1	A Beginner’s Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans2	A Beginner’s Guide, Java: The Complete Reference, Java For Dummies
Trans3	A Beginner’s Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans4	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition , Beginning Programming with Java,
Trans5	Android Programming: The Big Nerd Ranch, Beginning Programming with Java, Java 8 Pocket Guide
Trans6	A Beginner’s Guide, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans7	A Beginner’s Guide, Head First Java 2nd Edition , Beginning Programming with Java
Trans8	Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch,
Trans9	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition , Beginning Programming with Java,
Trans10	Beginning Programming with Java, Java 8 Pocket Guide, C++ Programming in Easy Steps
Trans11	A Beginner’s Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch

Trans12	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, HTML and CSS: Design and Build Websites
Trans13	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Java 8 Pocket Guide, HTML and CSS: Design and Build Websites
Trans14	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans15	Java For Dummies, Android Programming: The Big Nerd Ranch
Trans16	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans17	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans18	Head First Java 2nd Edition , Beginning Programming with Java, Java 8 Pocket Guide
Trans19	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans20	A Beginner's Guide, Java: The Complete Reference, Java For Dummies

Table 2 Amazon Data Sets and Transactions

2) Best Buy

Item #	Item Name
1	Digital Camera
2	Lab Top
3	Desk Top
4	Printer
5	Flash Drive
6	Microsoft Office
7	Speakers
8	Lab Top Case
9	Anti-Virus
10	External Hard-Drive

Table 3 Best Buy Items Names

Transaction ID	Transaction
Trans1	Desk Top, Printer, Flash Drive, Microsoft Office, Speakers, Anti-Virus
Trans2	Lab Top, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus
Trans3	Lab Top, Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive
Trans4	Lab Top, Printer, Flash Drive, Anti-Virus, External Hard-Drive, Lab Top Case
Trans5	Lab Top, Flash Drive, Lab Top Case, Anti-Virus
Trans6	Lab Top, Printer, Flash Drive, Microsoft Office
Trans7	Desk Top, Printer, Flash Drive, Microsoft Office
Trans8	Lab Top, External Hard-Drive, Anti-Virus
Trans9	Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, Speakers, External Hard-Drive
Trans10	Digital Camera , Lab Top, Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, External Hard-Drive, Speakers
Trans11	Lab Top, Desk Top, Lab Top Case, External Hard-Drive, Speakers, Anti-Virus
Trans12	Digital Camera , Lab Top, Lab Top Case, External Hard-Drive, Anti-Virus, Speakers
Trans13	Digital Camera , Speakers
Trans14	Digital Camera , Desk Top, Printer, Flash Drive, Microsoft Office
Trans15	Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, Speakers, External Hard-Drive
Trans16	Digital Camera, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive, Speakers

Trans17	Digital Camera , Lab Top, Lab Top Case
Trans18	Digital Camera , Lab Top Case, Speakers
Trans19	Digital Camera , Lab Top, Printer, Flash Drive, Microsoft Office, Speakers, Lab Top Case, Anti-Virus
Trans20	Digital Camera , Lab Top, Speakers, Anti-Virus, Lab Top Case

Table 4 Best Buy Data Sets Transactions

3) K-mart

Item #	Item Name
1	Quilts
2	Bedspreads
3	Decorative Pillows
4	Bed Skirts
5	Sheets
6	Shams
7	Bedding Collections
8	Kids Bedding
9	Embroidered Bedspread
10	Towels

Table 5 K-Mart data items

Transaction ID	Transaction
Trans1	Decorative Pillows, Quilts, Embroidered Bedspread
Trans2	Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections, Bed Skirts, Bedspreads, Sheets
Trans3	Decorative Pillows, Quilts, Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections
Trans4	Kids Bedding, Bedding Collections, Sheets, Bedspreads, Bed Skirts
Trans5	Decorative Pillows, Kids Bedding, Bedding Collections, Sheets, Bed Skirts, Bedspreads
Trans6	Bedding Collections, Bedspreads, Bed Skirts, Sheets, Shams, Kids Bedding
Trans7	Decorative Pillows, Quilts
Trans8	Decorative Pillows, Quilts, Embroidered Bedspread
Trans9	Bedspreads, Bed Skirts, Shams, Kids Bedding, Sheets
Trans10	Quilts, Embroidered Bedspread, Bedding Collections
Trans11	Bedding Collections, Bedspreads, Bed Skirts, Kids Bedding, Shams, Sheets
Trans12	Decorative Pillows, Quilts
Trans13	Embroidered Bedspread, Shams
Trans14	Sheets, Shams, Bed Skirts, Kids Bedding
Trans15	Decorative Pillows, Quilts
Trans16	Decorative Pillows, Kids Bedding, Bed Skirts, Shams
Trans17	Decorative Pillows, Shams, Bed Skirts
Trans18	Quilts, Sheets, Kids Bedding
Trans19	Shams, Bed Skirts, Kids Bedding, Sheets
Trans20	Decorative Pillows, Bedspreads, Shams, Sheets, Bed Skirts, Kids Bedding

Table 6 K-Mart Data Sets Transactions

4) Nike

Item #	Item Name
1	Running Shoe
2	Soccer Shoe
3	Socks
4	Swimming Shirt
5	Dry Fit V-Nick
6	Rash Guard
7	Sweatshirts
8	Hoodies
9	Tech Pants
10	Modern Pants

Table 7 Nike Data Items

Transaction ID	Transaction
Trans1	Running Shoe, Socks, Sweatshirts, Modern Pants
Trans2	Running Shoe, Socks, Sweatshirts
Trans3	Running Shoe, Socks, Sweatshirts, Modern Pants
Trans4	Running Shoe, Sweatshirts, Modern Pants
Trans5	Running Shoe, Socks, Sweatshirts, Modern Pants, Soccer Shoe
Trans6	Running Shoe, Socks, Sweatshirts
Trans7	Running Shoe, Socks, Sweatshirts, Modern Pants, Tech Pants, Rash Guard, Hoodies
Trans8	Swimming Shirt, Socks, Sweatshirts
Trans9	Swimming Shirt, Rash Guard, Dry Fit V-Nick, Hoodies, Tech Pants
Trans10	Swimming Shirt, Rash Guard, Dry
Trans11	Swimming Shirt, Rash Guard, Dry Fit V-Nick
Trans12	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Hoodies, Tech Pants, Dry Fit V-Nick
Trans13	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Tech Pants, Dry Fit V-Nick, Hoodies
Trans14	Running Shoe, Swimming Shirt, Rash Guard, Tech Pants, Hoodies, Dry Fit V-Nick
Trans15	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Dry Fit V-Nick, Rash Guard, Tech Pants
Trans16	Swimming Shirt, Soccer Shoe, Hoodies, Dry Fit V-Nick, Tech Pants, Rash Guard
Trans17	Running Shoe, Socks
Trans18	Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Rash Guard, Tech Pants, Dry Fit V-Nick
Trans19	Running Shoe, Swimming Shirt, Rash Guard
Trans20	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Tech Pants, Rash Guard, Dry Fit V-Nick

Table 8 Nike Data Sets Transactions

5) Generic

Item #	Item Name
1	A
2	B
3	C
4	D

5	E
6	F

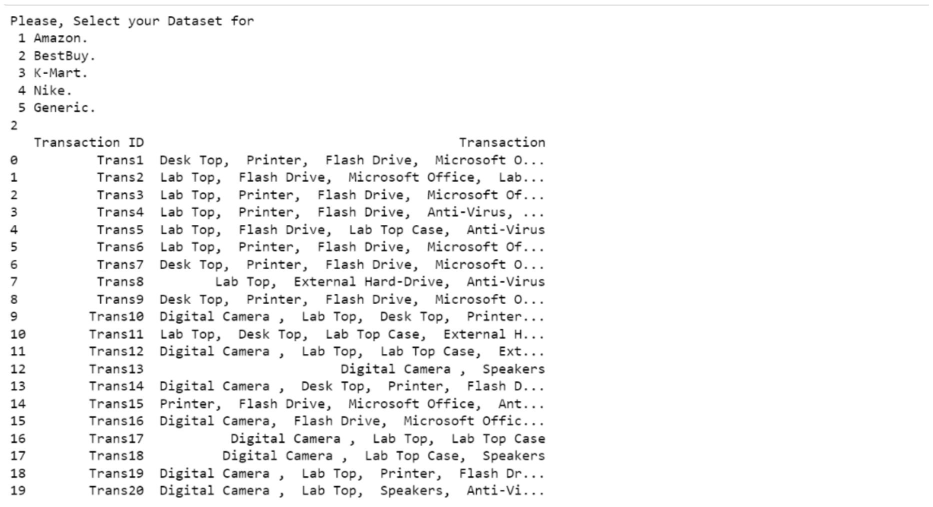
Table 9 Generic Items Names

Transaction ID	Transaction
Trans1	A, B, C
Trans2	A, B, C
Trans3	A, B, C, D
Trans4	A, B, C, D, E
Trans5	A, B, D, E
Trans6	A, D, E
Trans7	A, E
Trans8	A, E
Trans9	A, C, E
Trans10	A, C, E
Trans11	A, C, E

Table 10 Generic Data Sets and Transactions

```python
def load_and_display_dataset(choice):
    dataset_paths = {
        1: 'C:/Users/kanna/Desktop/Project/amazon.csv',
        2: 'C:/Users/kanna/Desktop/Project/bestbuy.csv',
        3: 'C:/Users/kanna/Desktop/Project/kmart.csv',
        4: 'C:/Users/kanna/Desktop/Project/nike.csv',
        5: 'C:/Users/kanna/Desktop/Project/generic.csv'
    }

    try:
        if choice in dataset_paths:
            df = pd.read_csv(dataset_paths[choice])
            return df
        else:
            print("Invalid choice. Please select a number between 1 and 5.")
            return None
    except FileNotFoundError:
        print(f"File not found for choice {choice}. Please check the file path and try again.")
        return None

try:
    choice = int(input("Please, Select your Dataset for \n 1 Amazon.\n 2 BestBuy.\n 3 K-Mart.\n 4 Nike.\n 5 Generic. \n"))
    df = load_and_display_dataset(choice)
    if df is not None:
        print(df)
except ValueError:
    print("Please enter a valid integer.")
```

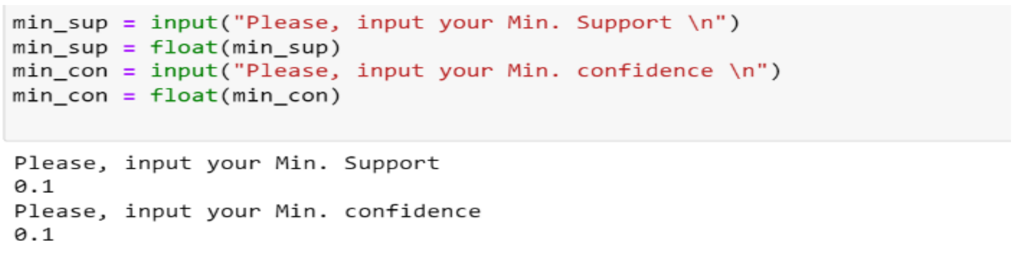
If we select 2 as our data, then the result will be.



```
Please, Select your Dataset for
 1 Amazon.
 2 BestBuy.
 3 K-Mart.
 4 Nike.
 5 Generic.
2
   Transaction ID                                        Transaction
0          Trans1  Desk Top,  Printer,  Flash Drive,  Microsoft O...
1          Trans2  Lab Top,  Flash Drive,  Microsoft Office,  Lab...
2          Trans3  Lab Top,  Printer,  Flash Drive,  Microsoft Of...
3          Trans4  Lab Top,  Printer,  Flash Drive,  Anti-Virus, ...
4          Trans5  Lab Top,  Flash Drive,  Lab Top Case,  Anti-Virus
5          Trans6  Lab Top,  Printer,  Flash Drive,  Microsoft Of...
6          Trans7  Desk Top,  Printer,  Flash Drive,  Microsoft O...
7          Trans8         Lab Top,  External Hard-Drive,  Anti-Virus
8          Trans9  Desk Top,  Printer,  Flash Drive,  Microsoft O...
9         Trans10  Digital Camera ,  Lab Top,  Desk Top,  Printer...
10        Trans11  Lab Top,  Desk Top,  Lab Top Case,  External H...
11        Trans12  Digital Camera ,  Lab Top,  Lab Top Case,  Ext...
12        Trans13                           Digital Camera ,  Speakers
13        Trans14  Digital Camera ,  Desk Top,  Printer,  Flash D...
14        Trans15  Printer,  Flash Drive,  Microsoft Office,  Ant...
15        Trans16  Digital Camera,  Flash Drive,  Microsoft Offic...
16        Trans17           Digital Camera ,  Lab Top,  Lab Top Case
17        Trans18          Digital Camera ,  Lab Top Case,  Speakers
18        Trans19  Digital Camera ,  Lab Top,  Printer,  Flash Dr...
19        Trans20  Digital Camera ,  Lab Top,  Speakers,  Anti-Vi...
```

3. User Inputs for Minimum Support and Confidence

The user is prompted to enter minimum support and confidence levels for the analysis. These thresholds are critical in determining the frequency and strength of the association rules derived from the data. Minimum support defines the lowest level of frequency an itemset must have to be considered relevant, while minimum confidence indicates the reliability of the inferred rules.

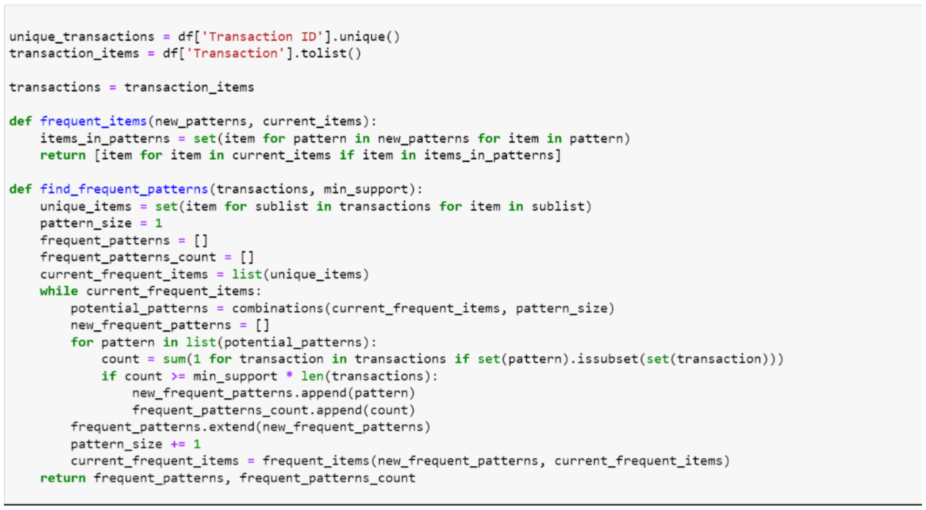


```
min_sup = input("Please, input your Min. Support \n")
min_sup = float(min_sup)
min_con = input("Please, input your Min. confidence \n")
min_con = float(min_con)
```

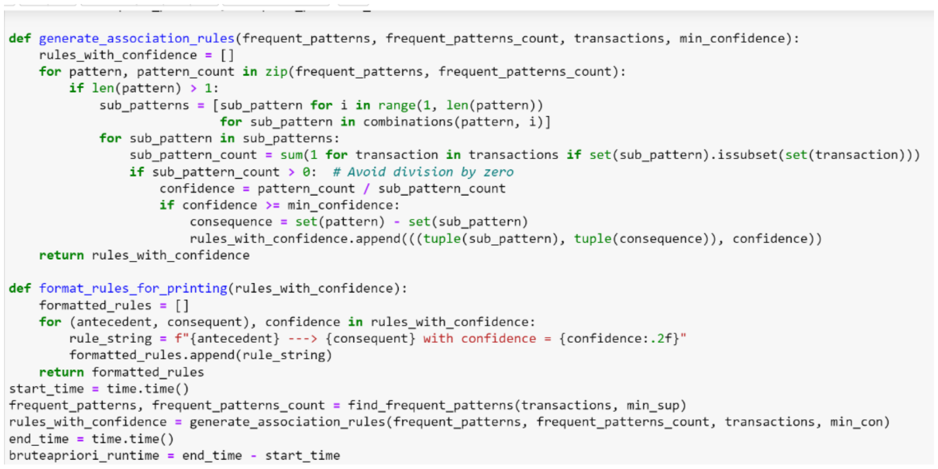
```
Please, input your Min. Support
0.1
Please, input your Min. confidence
0.1
```

4. Brute-Forced Apriori Implementation

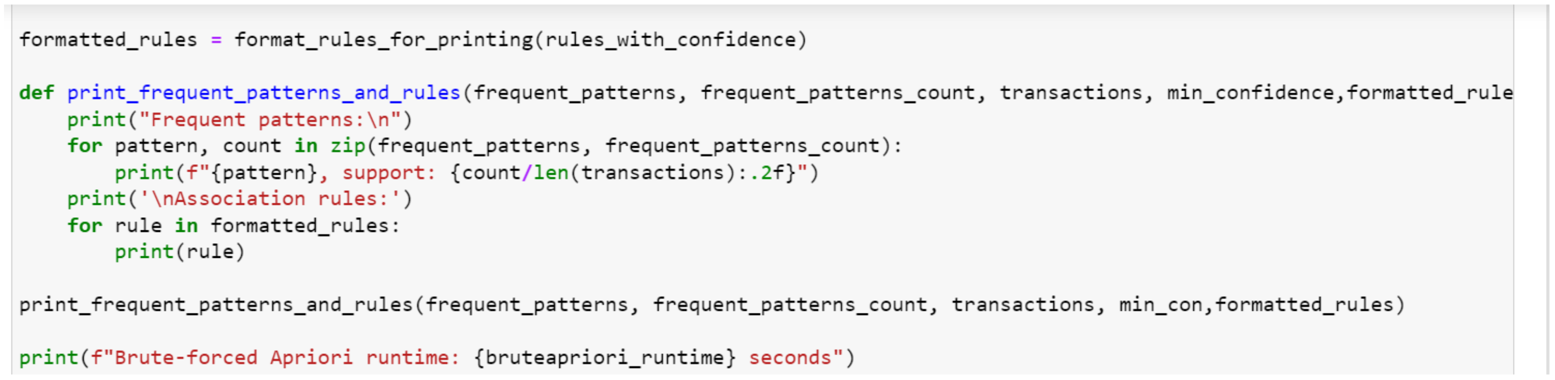
The brute-forced Apriori implementation involves manually processing the transaction data to identify frequent itemsets and generate association rules based on the user-defined minimum support and confidence. This section outlines the step-by-step process of the algorithm, including the preprocessing of transactions, computation of frequent itemsets, and rule generation.



```python
unique_transactions = df['Transaction ID'].unique()
transaction_items = df['Transaction'].tolist()

transactions = transaction_items

def frequent_items(new_patterns, current_items):
    items_in_patterns = set(item for pattern in new_patterns for item in pattern)
    return [item for item in current_items if item in items_in_patterns]

def find_frequent_patterns(transactions, min_support):
    unique_items = set(item for sublist in transactions for item in sublist)
    pattern_size = 1
    frequent_patterns = []
    frequent_patterns_count = []
    current_frequent_items = list(unique_items)
    while current_frequent_items:
        potential_patterns = combinations(current_frequent_items, pattern_size)
        new_frequent_patterns = []
        for pattern in list(potential_patterns):
            count = sum(1 for transaction in transactions if set(pattern).issubset(set(transaction)))
            if count >= min_support * len(transactions):
                new_frequent_patterns.append(pattern)
                frequent_patterns_count.append(count)
        frequent_patterns.extend(new_frequent_patterns)
        pattern_size += 1
        current_frequent_items = frequent_items(new_frequent_patterns, current_frequent_items)
    return frequent_patterns, frequent_patterns_count
```

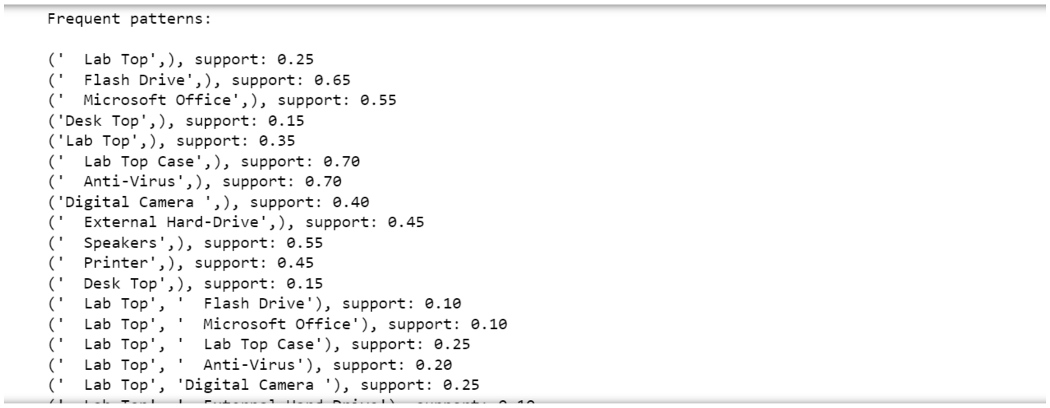


```python
def generate_association_rules(frequent_patterns, frequent_patterns_count, transactions, min_confidence):
    rules_with_confidence = []
    for pattern, pattern_count in zip(frequent_patterns, frequent_patterns_count):
        if len(pattern) > 1:
            sub_patterns = [sub_pattern for i in range(1, len(pattern))
                            for sub_pattern in combinations(pattern, i)]
            for sub_pattern in sub_patterns:
                sub_pattern_count = sum(1 for transaction in transactions if set(sub_pattern).issubset(set(transaction)))
                if sub_pattern_count > 0:  # Avoid division by zero
                    confidence = pattern_count / sub_pattern_count
                    if confidence >= min_confidence:
                        consequence = set(pattern) - set(sub_pattern)
                        rules_with_confidence.append(((tuple(sub_pattern), tuple(consequence)), confidence))
    return rules_with_confidence

def format_rules_for_printing(rules_with_confidence):
    formatted_rules = []
    for (antecedent, consequent), confidence in rules_with_confidence:
        rule_string = f"{antecedent} ---> {consequent} with confidence = {confidence:.2f}"
        formatted_rules.append(rule_string)
    return formatted_rules
start_time = time.time()
frequent_patterns, frequent_patterns_count = find_frequent_patterns(transactions, min_sup)
rules_with_confidence = generate_association_rules(frequent_patterns, frequent_patterns_count, transactions, min_con)
end_time = time.time()
bruteapriori_runtime = end_time - start_time
```

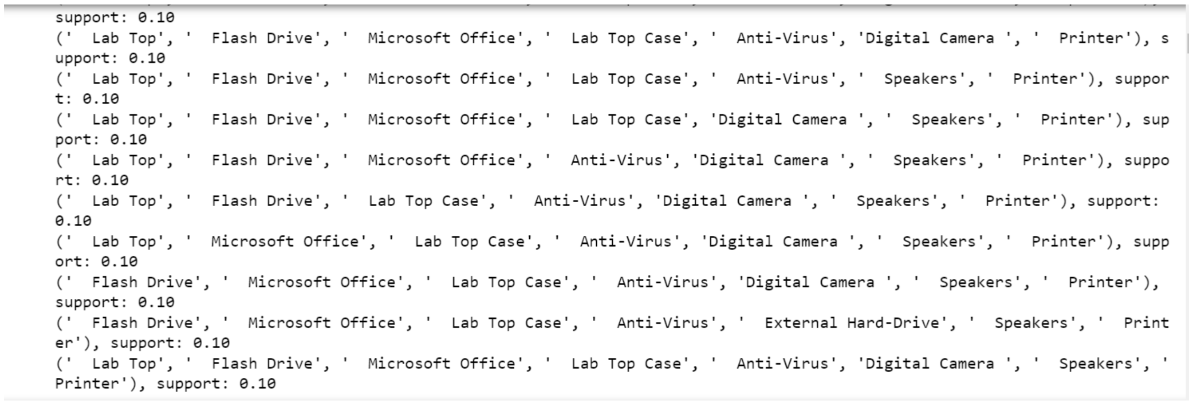


```python
formatted_rules = format_rules_for_printing(rules_with_confidence)

def print_frequent_patterns_and_rules(frequent_patterns, frequent_patterns_count, transactions, min_confidence,formatted_rule
    print("Frequent patterns:\n")
    for pattern, count in zip(frequent_patterns, frequent_patterns_count):
        print(f"{pattern}, support: {count/len(transactions):.2f}")
    print('\nAssociation rules:')
    for rule in formatted_rules:
        print(rule)

print_frequent_patterns_and_rules(frequent_patterns, frequent_patterns_count, transactions, min_con,formatted_rules)

print(f"Brute-forced Apriori runtime: {bruteapriori_runtime} seconds")
```

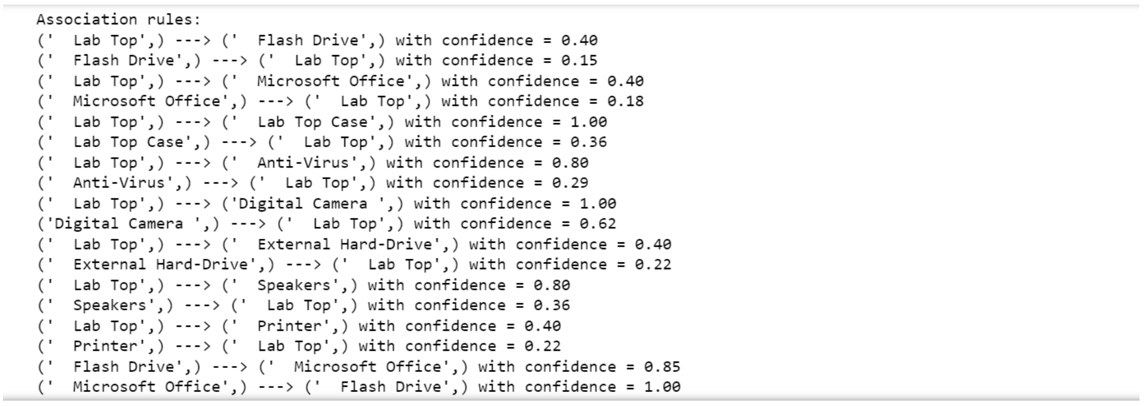
Results:



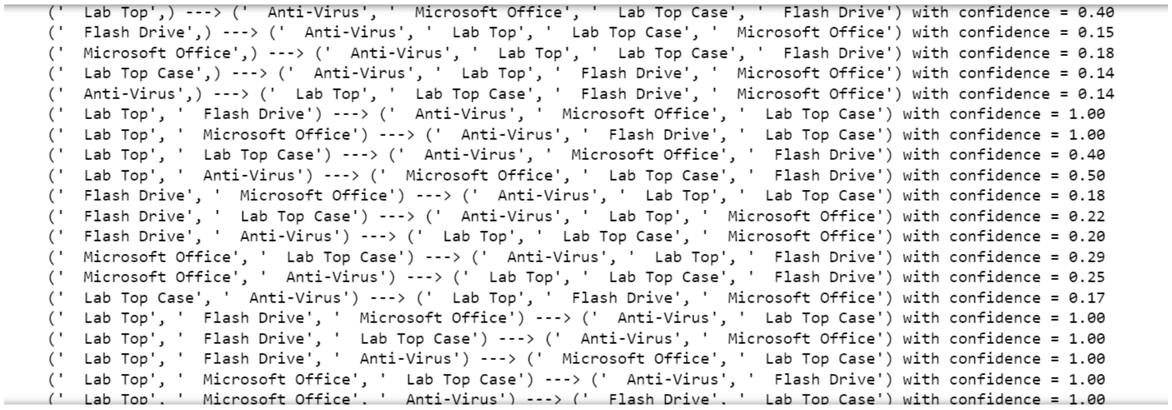
```
Frequent patterns:

('  Lab Top',), support: 0.25
('  Flash Drive',), support: 0.65
('  Microsoft Office',), support: 0.55
('Desk Top',), support: 0.15
('Lab Top',), support: 0.35
('  Lab Top Case',), support: 0.70
('  Anti-Virus',), support: 0.70
('Digital Camera ',), support: 0.40
('  External Hard-Drive',), support: 0.45
('  Speakers',), support: 0.55
('  Printer',), support: 0.45
('  Desk Top',), support: 0.15
('  Lab Top', '  Flash Drive'), support: 0.10
('  Lab Top', '  Microsoft Office'), support: 0.10
('  Lab Top', '  Lab Top Case'), support: 0.25
('  Lab Top', '  Anti-Virus'), support: 0.20
('  Lab Top', 'Digital Camera '), support: 0.25
```



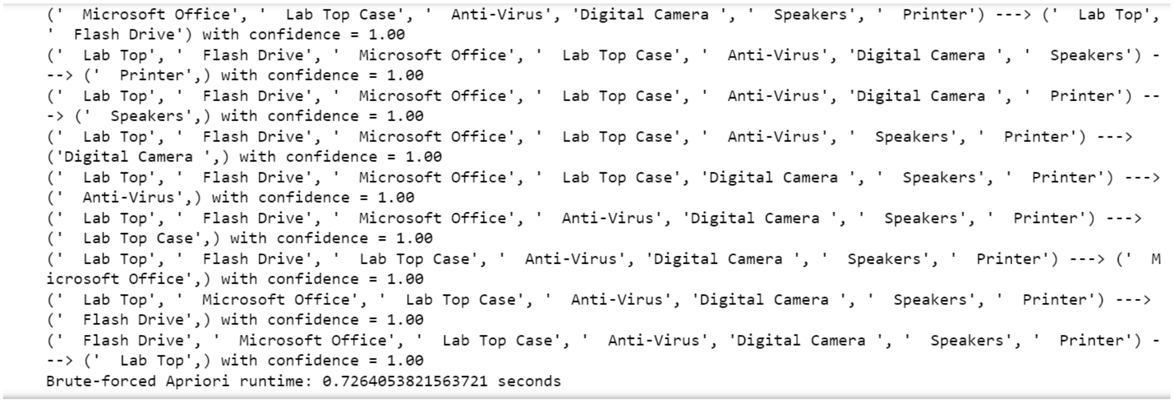
```
support: 0.10
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Printer'), s
upport: 0.10
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', '  Speakers', '  Printer'), suppor
t: 0.10
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', 'Digital Camera ', '  Speakers', '  Printer'), sup
port: 0.10
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer'), suppo
rt: 0.10
('  Lab Top', '  Flash Drive', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer'), support:
0.10
('  Lab Top', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer'), supp
ort: 0.10
('  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer'),
support: 0.10
('  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', '  External Hard-Drive', '  Speakers', '  Print
er'), support: 0.10
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '
Printer'), support: 0.10
```



```
Association rules:
('  Lab Top',) ---> ('  Flash Drive',) with confidence = 0.40
('  Flash Drive',) ---> ('  Lab Top',) with confidence = 0.15
('  Lab Top',) ---> ('  Microsoft Office',) with confidence = 0.40
('  Microsoft Office',) ---> ('  Lab Top',) with confidence = 0.18
('  Lab Top',) ---> ('  Lab Top Case',) with confidence = 1.00
('  Lab Top Case',) ---> ('  Lab Top',) with confidence = 0.36
('  Lab Top',) ---> ('  Anti-Virus',) with confidence = 0.80
('  Anti-Virus',) ---> ('  Lab Top',) with confidence = 0.29
('  Lab Top',) ---> ('Digital Camera ',) with confidence = 1.00
('Digital Camera ',) ---> ('  Lab Top',) with confidence = 0.62
('  Lab Top',) ---> ('  External Hard-Drive',) with confidence = 0.40
('  External Hard-Drive',) ---> ('  Lab Top',) with confidence = 0.22
('  Lab Top',) ---> ('  Speakers',) with confidence = 0.80
('  Speakers',) ---> ('  Lab Top',) with confidence = 0.36
('  Lab Top',) ---> ('  Printer',) with confidence = 0.40
('  Printer',) ---> ('  Lab Top',) with confidence = 0.22
('  Flash Drive',) ---> ('  Microsoft Office',) with confidence = 0.85
('  Microsoft Office',) ---> ('  Flash Drive',) with confidence = 1.00
```



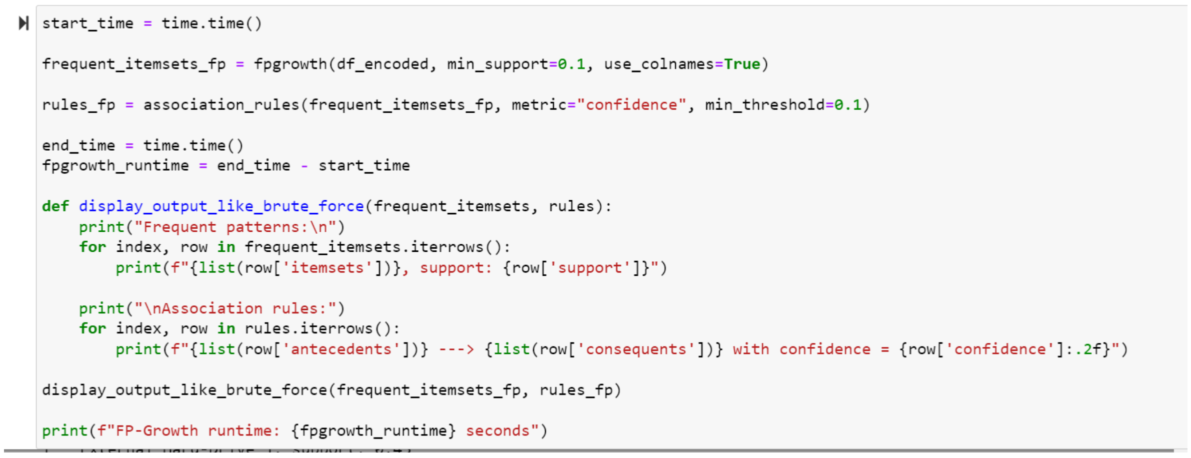
```
('  Lab Top',) ---> ('  Anti-Virus', '  Microsoft Office', '  Lab Top Case', '  Flash Drive') with confidence = 0.40
('  Flash Drive',) ---> ('  Anti-Virus', '  Lab Top', '  Lab Top Case', '  Microsoft Office') with confidence = 0.15
('  Microsoft Office',) ---> ('  Anti-Virus', '  Lab Top', '  Lab Top Case', '  Flash Drive') with confidence = 0.18
('  Lab Top Case',) ---> ('  Anti-Virus', '  Lab Top', '  Flash Drive', '  Microsoft Office') with confidence = 0.14
('  Anti-Virus',) ---> ('  Lab Top', '  Lab Top Case', '  Flash Drive', '  Microsoft Office') with confidence = 0.14
('  Lab Top', '  Flash Drive') ---> ('  Anti-Virus', '  Microsoft Office', '  Lab Top Case') with confidence = 1.00
('  Lab Top', '  Microsoft Office') ---> ('  Anti-Virus', '  Flash Drive', '  Lab Top Case') with confidence = 1.00
('  Lab Top', '  Lab Top Case') ---> ('  Anti-Virus', '  Microsoft Office', '  Flash Drive') with confidence = 0.40
('  Lab Top', '  Anti-Virus') ---> ('  Microsoft Office', '  Lab Top Case', '  Flash Drive') with confidence = 0.50
('  Flash Drive', '  Microsoft Office') ---> ('  Anti-Virus', '  Lab Top', '  Lab Top Case') with confidence = 0.18
('  Flash Drive', '  Lab Top Case') ---> ('  Anti-Virus', '  Lab Top', '  Microsoft Office') with confidence = 0.22
('  Flash Drive', '  Anti-Virus') ---> ('  Lab Top', '  Lab Top Case', '  Microsoft Office') with confidence = 0.20
('  Microsoft Office', '  Lab Top Case') ---> ('  Anti-Virus', '  Lab Top', '  Flash Drive') with confidence = 0.29
('  Microsoft Office', '  Anti-Virus') ---> ('  Lab Top', '  Lab Top Case', '  Flash Drive') with confidence = 0.25
('  Lab Top Case', '  Anti-Virus') ---> ('  Lab Top', '  Flash Drive', '  Microsoft Office') with confidence = 0.17
('  Lab Top', '  Flash Drive', '  Microsoft Office') ---> ('  Anti-Virus', '  Lab Top Case') with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Lab Top Case') ---> ('  Anti-Virus', '  Microsoft Office') with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Anti-Virus') ---> ('  Microsoft Office', '  Lab Top Case') with confidence = 1.00
('  Lab Top', '  Microsoft Office', '  Lab Top Case') ---> ('  Anti-Virus', '  Flash Drive') with confidence = 1.00
('  Lab Top', '  Microsoft Office', '  Anti-Virus') ---> ('  Flash Drive', '  Lab Top Case') with confidence = 1.00
```



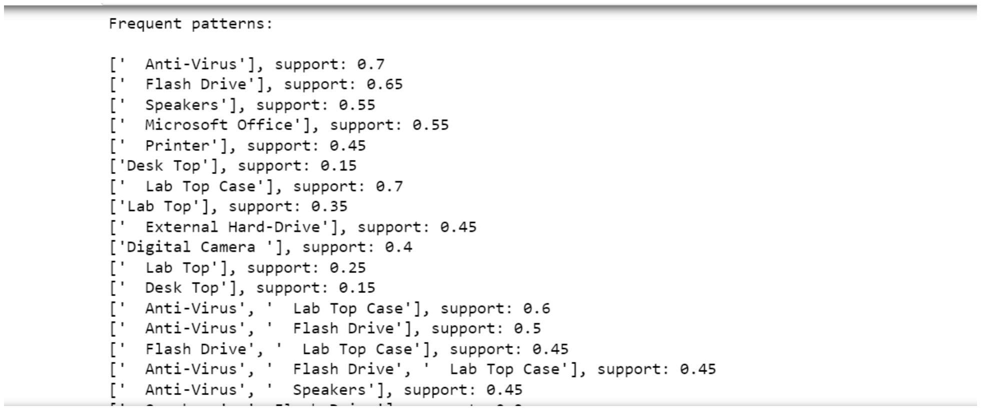
```
('  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer') ---> ('  Lab Top',
'  Flash Drive') with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers') -
--> ('  Printer',) with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Printer') --
-> ('  Speakers',) with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', '  Speakers', '  Printer') --->
('Digital Camera ',) with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Lab Top Case', 'Digital Camera ', '  Speakers', '  Printer') --->
('  Anti-Virus',) with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Microsoft Office', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer') --->
('  Lab Top Case',) with confidence = 1.00
('  Lab Top', '  Flash Drive', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer') ---> ('  M
icrosoft Office',) with confidence = 1.00
('  Lab Top', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer') --->
('  Flash Drive',) with confidence = 1.00
('  Flash Drive', '  Microsoft Office', '  Lab Top Case', '  Anti-Virus', 'Digital Camera ', '  Speakers', '  Printer') -
--> ('  Lab Top',) with confidence = 1.00
Brute-forced Apriori runtime: 0.7264053821563721 seconds
```

5. Validation with Python Package of Apriori

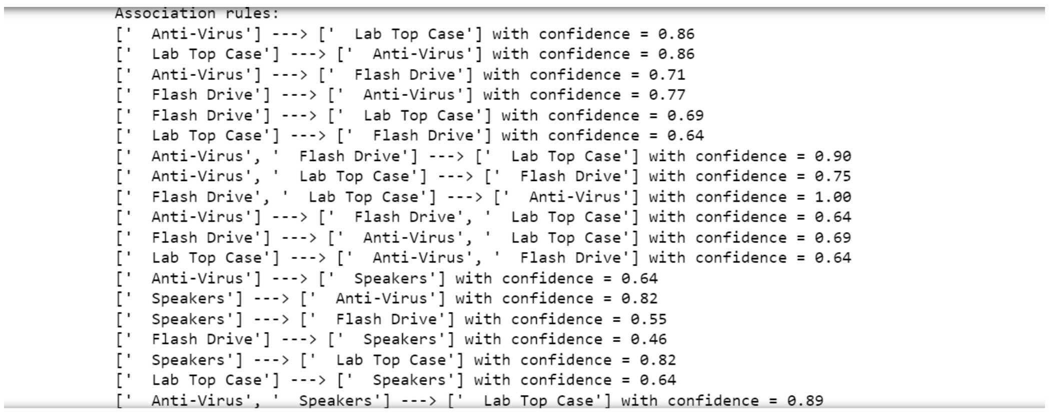
To validate the results obtained from the brute-forced approach, the 'mlxtend' library's Apriori function is used. This section compares the performance and output of the manual implementation with the package-based approach, highlighting the efficiency and accuracy of using a well-optimized library for association rule learning.



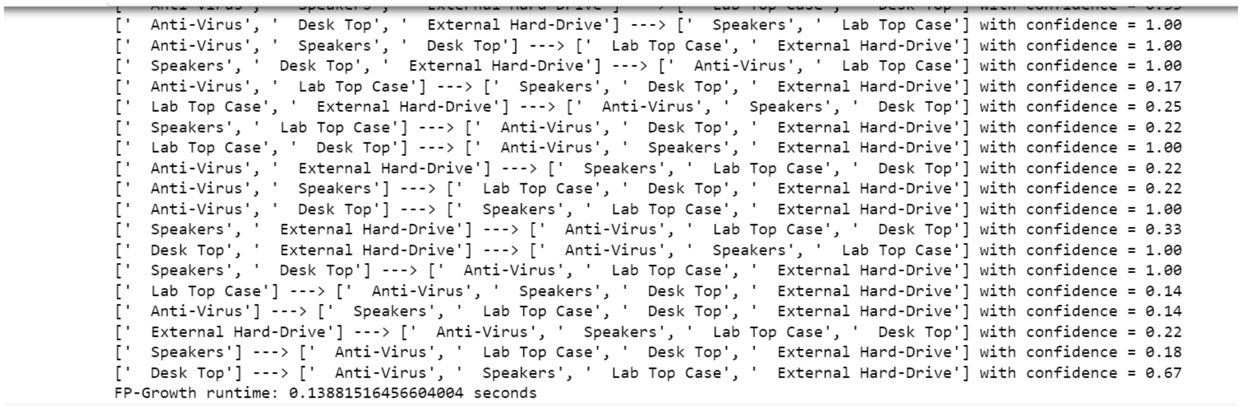
```
start_time = time.time()

frequent_itemsets_fp = fpgrowth(df_encoded, min_support=0.1, use_colnames=True)

rules_fp = association_rules(frequent_itemsets_fp, metric="confidence", min_threshold=0.1)

end_time = time.time()
fpgrowth_runtime = end_time - start_time

def display_output_like_brute_force(frequent_itemsets, rules):
    print("Frequent patterns:\n")
    for index, row in frequent_itemsets.iterrows():
        print(f"{list(row['itemsets'])}, support: {row['support']}")

    print("\nAssociation rules:")
    for index, row in rules.iterrows():
        print(f"{list(row['antecedents'])} ---> {list(row['consequents'])} with confidence = {row['confidence']:.2f}")

display_output_like_brute_force(frequent_itemsets_fp, rules_fp)

print(f"FP-Growth runtime: {fpgrowth_runtime} seconds")
```



```
Frequent patterns:

[' Anti-Virus'], support: 0.7
[' Flash Drive'], support: 0.65
[' Speakers'], support: 0.55
[' Microsoft Office'], support: 0.55
[' Printer'], support: 0.45
['Desk Top'], support: 0.15
[' Lab Top Case'], support: 0.7
['Lab Top'], support: 0.35
[' External Hard-Drive'], support: 0.45
['Digital Camera '], support: 0.4
[' Lab Top'], support: 0.25
[' Desk Top'], support: 0.15
[' Anti-Virus', ' Lab Top Case'], support: 0.6
[' Anti-Virus', ' Flash Drive'], support: 0.5
[' Flash Drive', ' Lab Top Case'], support: 0.45
[' Anti-Virus', ' Flash Drive', ' Lab Top Case'], support: 0.45
[' Anti-Virus', ' Speakers'], support: 0.45
```



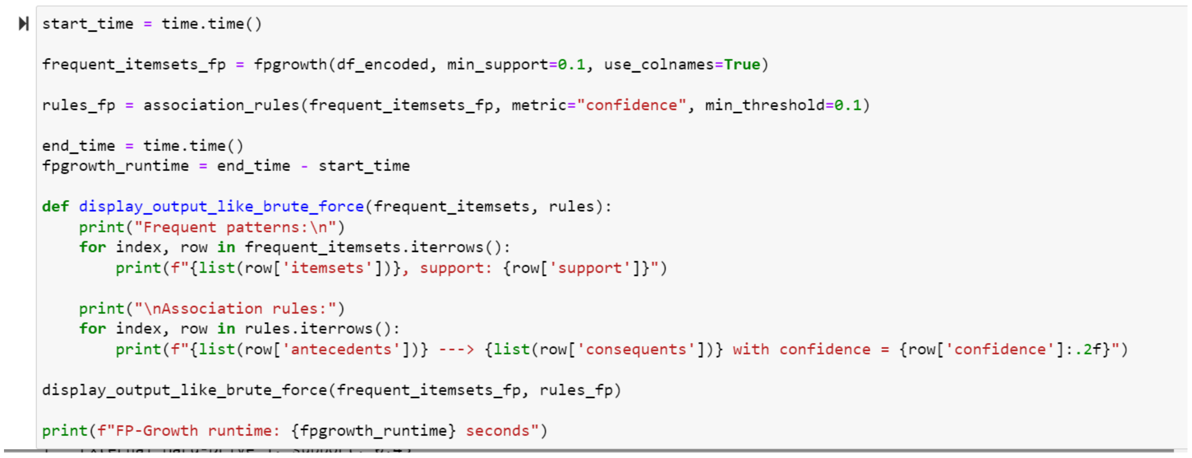
```
Association rules:
[' Anti-Virus'] ---> [' Lab Top Case'] with confidence = 0.86
[' Lab Top Case'] ---> [' Anti-Virus'] with confidence = 0.86
[' Anti-Virus'] ---> [' Flash Drive'] with confidence = 0.71
[' Flash Drive'] ---> [' Anti-Virus'] with confidence = 0.77
[' Flash Drive'] ---> [' Lab Top Case'] with confidence = 0.69
[' Lab Top Case'] ---> [' Flash Drive'] with confidence = 0.64
[' Anti-Virus', ' Flash Drive'] ---> [' Lab Top Case'] with confidence = 0.90
[' Anti-Virus', ' Lab Top Case'] ---> [' Flash Drive'] with confidence = 0.75
[' Flash Drive', ' Lab Top Case'] ---> [' Anti-Virus'] with confidence = 1.00
[' Anti-Virus'] ---> [' Flash Drive', ' Lab Top Case'] with confidence = 0.64
[' Flash Drive'] ---> [' Anti-Virus', ' Lab Top Case'] with confidence = 0.69
[' Lab Top Case'] ---> [' Anti-Virus', ' Flash Drive'] with confidence = 0.64
[' Anti-Virus'] ---> [' Speakers'] with confidence = 0.64
[' Speakers'] ---> [' Anti-Virus'] with confidence = 0.82
[' Speakers'] ---> [' Flash Drive'] with confidence = 0.55
[' Flash Drive'] ---> [' Speakers'] with confidence = 0.46
[' Speakers'] ---> [' Lab Top Case'] with confidence = 0.82
[' Lab Top Case'] ---> [' Speakers'] with confidence = 0.64
[' Anti-Virus', ' Speakers'] ---> [' Lab Top Case'] with confidence = 0.89
```



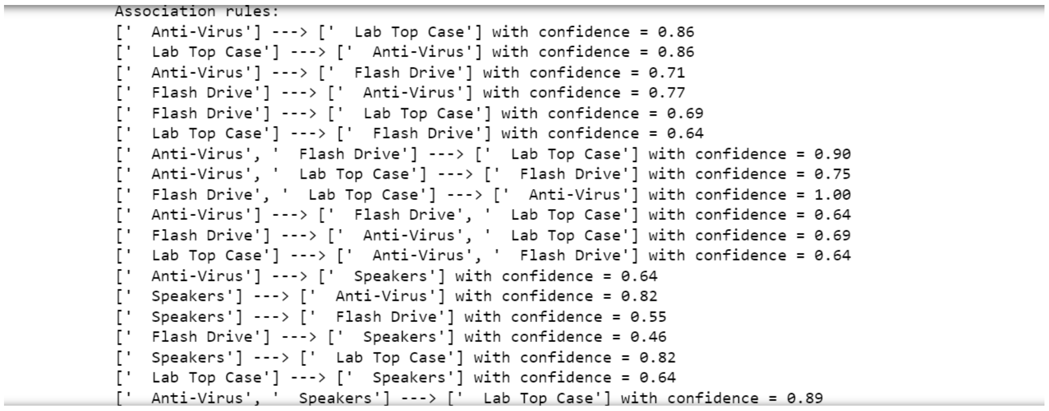
```
[' Anti-Virus', ' Desk Top', ' External Hard-Drive'] ---> [' Speakers', ' Lab Top Case'] with confidence = 1.00
[' Anti-Virus', ' Speakers', ' Desk Top'] ---> [' Lab Top Case', ' External Hard-Drive'] with confidence = 1.00
[' Speakers', ' Desk Top', ' External Hard-Drive'] ---> [' Anti-Virus', ' Lab Top Case'] with confidence = 1.00
[' Anti-Virus', ' Lab Top Case'] ---> [' Speakers', ' Desk Top', ' External Hard-Drive'] with confidence = 0.17
[' Lab Top Case', ' External Hard-Drive'] ---> [' Anti-Virus', ' Speakers', ' Desk Top'] with confidence = 0.25
[' Speakers', ' Lab Top Case'] ---> [' Anti-Virus', ' Desk Top', ' External Hard-Drive'] with confidence = 0.22
[' Lab Top Case', ' Desk Top'] ---> [' Anti-Virus', ' Speakers', ' External Hard-Drive'] with confidence = 1.00
[' Anti-Virus', ' External Hard-Drive'] ---> [' Speakers', ' Lab Top Case', ' Desk Top'] with confidence = 0.22
[' Anti-Virus', ' Speakers'] ---> [' Lab Top Case', ' Desk Top', ' External Hard-Drive'] with confidence = 0.22
[' Anti-Virus', ' Desk Top'] ---> [' Speakers', ' Lab Top Case', ' External Hard-Drive'] with confidence = 1.00
[' Speakers', ' External Hard-Drive'] ---> [' Anti-Virus', ' Lab Top Case', ' Desk Top'] with confidence = 0.33
[' Desk Top', ' External Hard-Drive'] ---> [' Anti-Virus', ' Speakers', ' Lab Top Case'] with confidence = 1.00
[' Speakers', ' Desk Top'] ---> [' Anti-Virus', ' Lab Top Case', ' External Hard-Drive'] with confidence = 1.00
[' Lab Top Case'] ---> [' Anti-Virus', ' Speakers', ' Desk Top', ' External Hard-Drive'] with confidence = 0.14
[' Anti-Virus'] ---> [' Speakers', ' Lab Top Case', ' Desk Top', ' External Hard-Drive'] with confidence = 0.14
[' External Hard-Drive'] ---> [' Anti-Virus', ' Speakers', ' Lab Top Case', ' Desk Top'] with confidence = 0.22
[' Speakers'] ---> [' Anti-Virus', ' Lab Top Case', ' Desk Top', ' External Hard-Drive'] with confidence = 0.18
[' Desk Top'] ---> [' Anti-Virus', ' Speakers', ' Lab Top Case', ' External Hard-Drive'] with confidence = 0.67
FP-Growth runtime: 0.13881516456604004 seconds
```

6. Validation with FP-Growth Algorithm

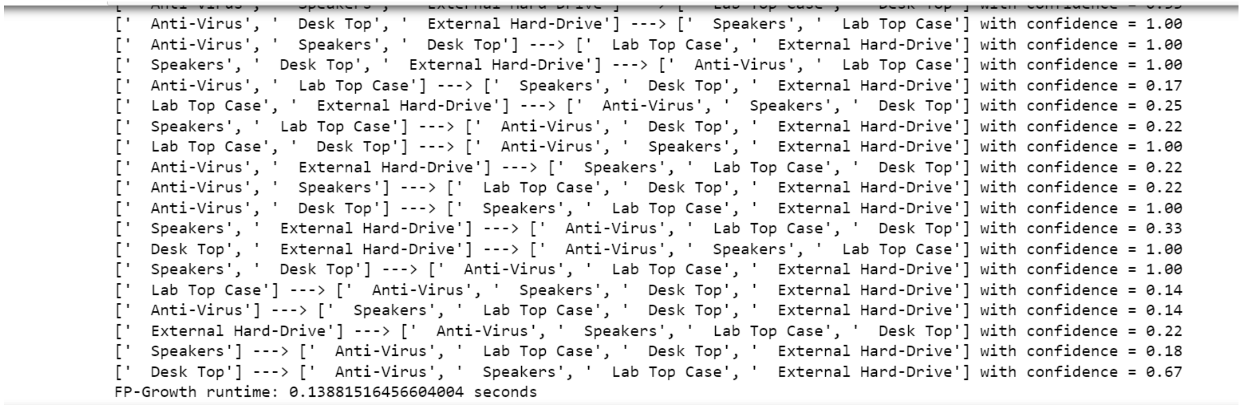
The FP-Growth algorithm offers an efficient alternative to Apriori for finding frequent itemsets without candidate generation. This section describes the application of FP-Growth, using the 'mlxtend' library, to the transaction data and compares its performance and findings with the Apriori implementations.



```python
start_time = time.time()

frequent_itemsets_fp = fpgrowth(df_encoded, min_support=0.1, use_colnames=True)

rules_fp = association_rules(frequent_itemsets_fp, metric="confidence", min_threshold=0.1)

end_time = time.time()
fpgrowth_runtime = end_time - start_time

def display_output_like_brute_force(frequent_itemsets, rules):
    print("Frequent patterns:\n")
    for index, row in frequent_itemsets.iterrows():
        print(f"{list(row['itemsets'])}, support: {row['support']}")

    print("\nAssociation rules:")
    for index, row in rules.iterrows():
        print(f"{list(row['antecedents'])} ---> {list(row['consequents'])} with confidence = {row['confidence']:.2f}")

display_output_like_brute_force(frequent_itemsets_fp, rules_fp)

print(f"FP-Growth runtime: {fpgrowth_runtime} seconds")
```



```
Association rules:
['  Anti-Virus'] ---> ['  Lab Top Case'] with confidence = 0.86
['  Lab Top Case'] ---> ['  Anti-Virus'] with confidence = 0.86
['  Anti-Virus'] ---> ['  Flash Drive'] with confidence = 0.71
['  Flash Drive'] ---> ['  Anti-Virus'] with confidence = 0.77
['  Flash Drive'] ---> ['  Lab Top Case'] with confidence = 0.69
['  Lab Top Case'] ---> ['  Flash Drive'] with confidence = 0.64
['  Anti-Virus', '  Flash Drive'] ---> ['  Lab Top Case'] with confidence = 0.90
['  Anti-Virus', '  Lab Top Case'] ---> ['  Flash Drive'] with confidence = 0.75
['  Flash Drive', '  Lab Top Case'] ---> ['  Anti-Virus'] with confidence = 1.00
['  Anti-Virus'] ---> ['  Flash Drive', '  Lab Top Case'] with confidence = 0.64
['  Flash Drive'] ---> ['  Anti-Virus', '  Lab Top Case'] with confidence = 0.69
['  Lab Top Case'] ---> ['  Anti-Virus', '  Flash Drive'] with confidence = 0.64
['  Anti-Virus'] ---> ['  Speakers'] with confidence = 0.64
['  Speakers'] ---> ['  Anti-Virus'] with confidence = 0.82
['  Speakers'] ---> ['  Flash Drive'] with confidence = 0.55
['  Flash Drive'] ---> ['  Speakers'] with confidence = 0.46
['  Speakers'] ---> ['  Lab Top Case'] with confidence = 0.82
['  Lab Top Case'] ---> ['  Speakers'] with confidence = 0.64
['  Anti-Virus', '  Speakers'] ---> ['  Lab Top Case'] with confidence = 0.89
```



```
['  Anti-Virus', '  Desk Top', '  External Hard-Drive'] ---> ['  Speakers', '  Lab Top Case'] with confidence = 1.00
['  Anti-Virus', '  Speakers', '  Desk Top'] ---> ['  Lab Top Case', '  External Hard-Drive'] with confidence = 1.00
['  Speakers', '  Desk Top', '  External Hard-Drive'] ---> ['  Anti-Virus', '  Lab Top Case'] with confidence = 1.00
['  Anti-Virus', '  Lab Top Case'] ---> ['  Speakers', '  Desk Top', '  External Hard-Drive'] with confidence = 0.17
['  Lab Top Case', '  External Hard-Drive'] ---> ['  Anti-Virus', '  Speakers', '  Desk Top'] with confidence = 0.25
['  Speakers', '  Lab Top Case'] ---> ['  Anti-Virus', '  Desk Top', '  External Hard-Drive'] with confidence = 0.22
['  Lab Top Case', '  Desk Top'] ---> ['  Anti-Virus', '  Speakers', '  External Hard-Drive'] with confidence = 1.00
['  Anti-Virus', '  External Hard-Drive'] ---> ['  Speakers', '  Lab Top Case', '  Desk Top'] with confidence = 0.22
['  Anti-Virus', '  Speakers'] ---> ['  Lab Top Case', '  Desk Top', '  External Hard-Drive'] with confidence = 0.22
['  Anti-Virus', '  Desk Top'] ---> ['  Speakers', '  Lab Top Case', '  External Hard-Drive'] with confidence = 1.00
['  Speakers', '  External Hard-Drive'] ---> ['  Anti-Virus', '  Lab Top Case', '  Desk Top'] with confidence = 0.33
['  Desk Top', '  External Hard-Drive'] ---> ['  Anti-Virus', '  Speakers', '  Lab Top Case'] with confidence = 1.00
['  Speakers', '  Desk Top'] ---> ['  Anti-Virus', '  Lab Top Case', '  External Hard-Drive'] with confidence = 1.00
['  Lab Top Case'] ---> ['  Anti-Virus', '  Speakers', '  Desk Top', '  External Hard-Drive'] with confidence = 0.14
['  Anti-Virus'] ---> ['  Speakers', '  Lab Top Case', '  Desk Top', '  External Hard-Drive'] with confidence = 0.14
['  External Hard-Drive'] ---> ['  Anti-Virus', '  Speakers', '  Lab Top Case', '  Desk Top'] with confidence = 0.22
['  Speakers'] ---> ['  Anti-Virus', '  Lab Top Case', '  Desk Top', '  External Hard-Drive'] with confidence = 0.18
['  Desk Top'] ---> ['  Anti-Virus', '  Speakers', '  Lab Top Case', '  External Hard-Drive'] with confidence = 0.67
FP-Growth runtime: 0.13881516456604004 seconds
```

7. Comparing time taken by each algorithm

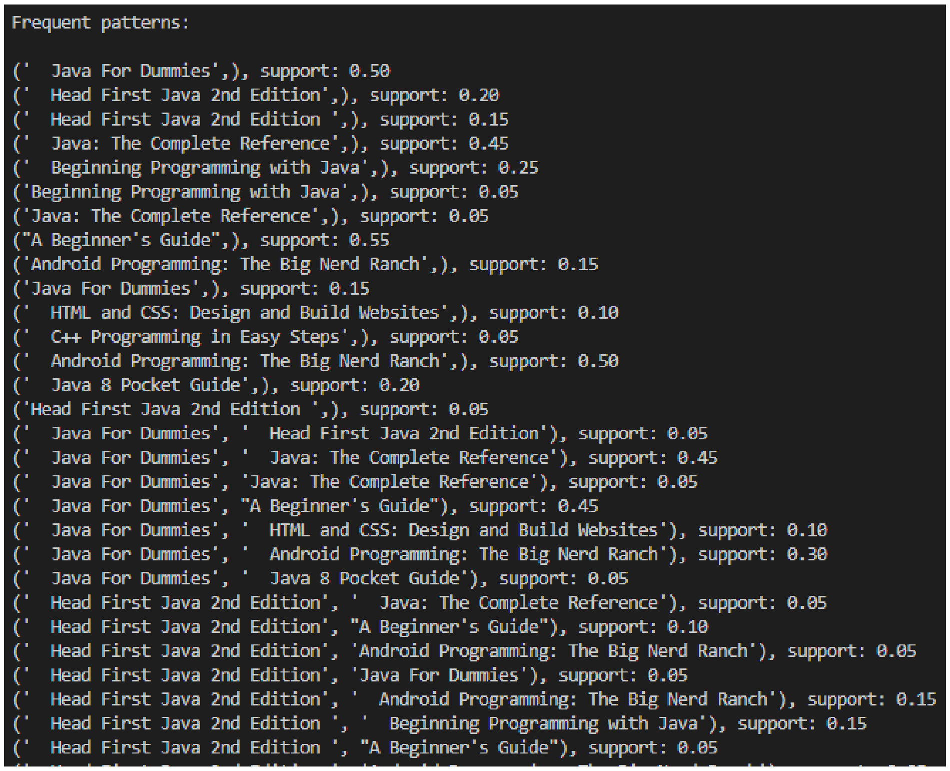
```
data = {
    "Algorithm": ["BruteApriori", "Apriori", "FPGrowth"],
    "Runtime": [bruteapriori_runtime, apriori_runtime, fpgrowth_runtime]
}

df = pd.DataFrame(data)
df_sorted = df.sort_values(by="Runtime", ascending=True)
print(df_sorted)
```

```
      Algorithm   Runtime
0  BruteApriori  0.003513
2      FPGrowth  0.013688
1       Apriori  0.016238
```

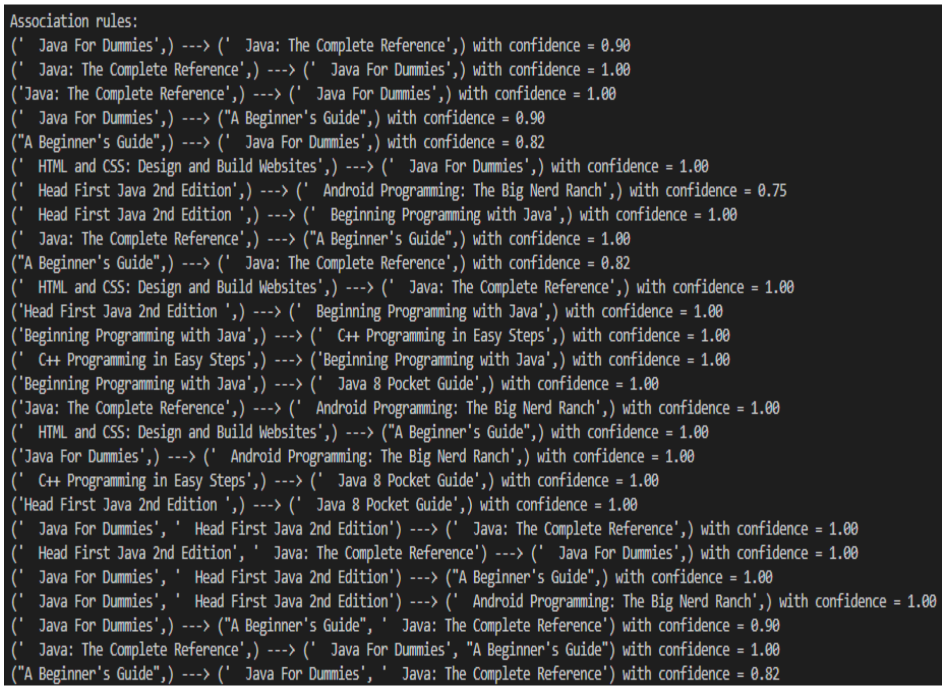
8. Results based on different minimum support and confidence on each data set.

For Amazon Data set:

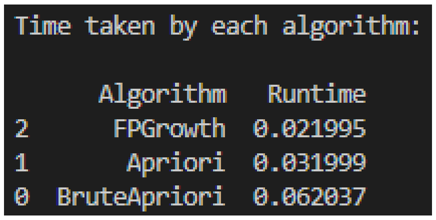
1. Support: 0.01 (1%), Confidence: 0.7 (70%)



```
Frequent patterns:

('  Java For Dummies',), support: 0.50
('  Head First Java 2nd Edition',), support: 0.20
('  Head First Java 2nd Edition ',), support: 0.15
('  Java: The Complete Reference',), support: 0.45
('  Beginning Programming with Java',), support: 0.25
('Beginning Programming with Java',), support: 0.05
('Java: The Complete Reference',), support: 0.05
("A Beginner's Guide",), support: 0.55
('Android Programming: The Big Nerd Ranch',), support: 0.15
('Java For Dummies',), support: 0.15
('  HTML and CSS: Design and Build Websites',), support: 0.10
('  C++ Programming in Easy Steps',), support: 0.05
('  Android Programming: The Big Nerd Ranch',), support: 0.50
('  Java 8 Pocket Guide',), support: 0.20
('Head First Java 2nd Edition ',), support: 0.05
('  Java For Dummies', '  Head First Java 2nd Edition'), support: 0.05
('  Java For Dummies', '  Java: The Complete Reference'), support: 0.45
('  Java For Dummies', 'Java: The Complete Reference'), support: 0.05
('  Java For Dummies', "A Beginner's Guide"), support: 0.45
('  Java For Dummies', '  HTML and CSS: Design and Build Websites'), support: 0.10
('  Java For Dummies', '  Android Programming: The Big Nerd Ranch'), support: 0.30
('  Java For Dummies', '  Java 8 Pocket Guide'), support: 0.05
('  Head First Java 2nd Edition', '  Java: The Complete Reference'), support: 0.05
('  Head First Java 2nd Edition', "A Beginner's Guide"), support: 0.10
('  Head First Java 2nd Edition', 'Android Programming: The Big Nerd Ranch'), support: 0.05
('  Head First Java 2nd Edition', 'Java For Dummies'), support: 0.05
('  Head First Java 2nd Edition', '  Android Programming: The Big Nerd Ranch'), support: 0.15
('  Head First Java 2nd Edition ', '  Beginning Programming with Java'), support: 0.15
('  Head First Java 2nd Edition ', "A Beginner's Guide"), support: 0.05
```

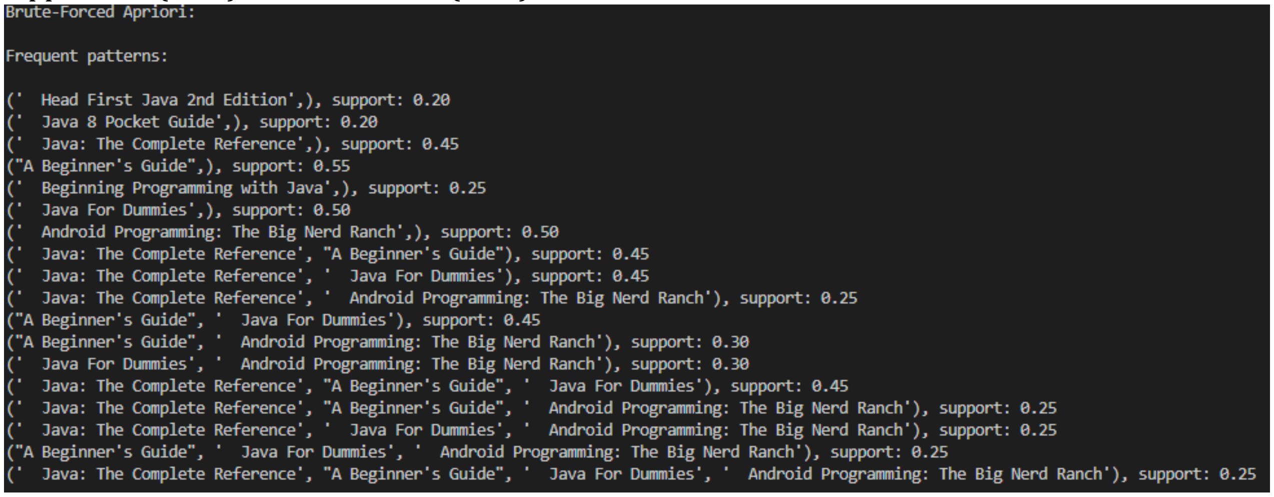


```
Association rules:
('  Java For Dummies',) ---> ('  Java: The Complete Reference',) with confidence = 0.90
('  Java: The Complete Reference',) ---> ('  Java For Dummies',) with confidence = 1.00
('Java: The Complete Reference',) ---> ('  Java For Dummies',) with confidence = 1.00
('  Java For Dummies',) ---> ("A Beginner's Guide",) with confidence = 0.90
("A Beginner's Guide",) ---> ('  Java For Dummies',) with confidence = 0.82
('  HTML and CSS: Design and Build Websites',) ---> ('  Java For Dummies',) with confidence = 1.00
('  Head First Java 2nd Edition',) ---> ('  Android Programming: The Big Nerd Ranch',) with confidence = 0.75
('  Head First Java 2nd Edition ',) ---> ('  Beginning Programming with Java',) with confidence = 1.00
('  Java: The Complete Reference',) ---> ("A Beginner's Guide",) with confidence = 1.00
("A Beginner's Guide",) ---> ('  Java: The Complete Reference',) with confidence = 0.82
('  HTML and CSS: Design and Build Websites',) ---> ('  Java: The Complete Reference',) with confidence = 1.00
('Head First Java 2nd Edition ',) ---> ('  Beginning Programming with Java',) with confidence = 1.00
('Beginning Programming with Java',) ---> ('  C++ Programming in Easy Steps',) with confidence = 1.00
('  C++ Programming in Easy Steps',) ---> ('Beginning Programming with Java',) with confidence = 1.00
('Beginning Programming with Java',) ---> ('  Java 8 Pocket Guide',) with confidence = 1.00
('Java: The Complete Reference',) ---> ('  Android Programming: The Big Nerd Ranch',) with confidence = 1.00
('  HTML and CSS: Design and Build Websites',) ---> ("A Beginner's Guide",) with confidence = 1.00
('Java For Dummies',) ---> ('  Android Programming: The Big Nerd Ranch',) with confidence = 1.00
('  C++ Programming in Easy Steps',) ---> ('  Java 8 Pocket Guide',) with confidence = 1.00
('Head First Java 2nd Edition ',) ---> ('  Java 8 Pocket Guide',) with confidence = 1.00
('  Java For Dummies', '  Head First Java 2nd Edition') ---> ('  Java: The Complete Reference',) with confidence = 1.00
('  Head First Java 2nd Edition', '  Java: The Complete Reference') ---> ('  Java For Dummies',) with confidence = 1.00
('  Java For Dummies', '  Head First Java 2nd Edition') ---> ("A Beginner's Guide",) with confidence = 1.00
('  Java For Dummies', '  Head First Java 2nd Edition') ---> ('  Android Programming: The Big Nerd Ranch',) with confidence = 1.00
('  Java For Dummies',) ---> ("A Beginner's Guide", '  Java: The Complete Reference') with confidence = 0.90
('  Java: The Complete Reference',) ---> ('  Java For Dummies', "A Beginner's Guide") with confidence = 1.00
("A Beginner's Guide",) ---> ('  Java For Dummies', '  Java: The Complete Reference') with confidence = 0.82
```

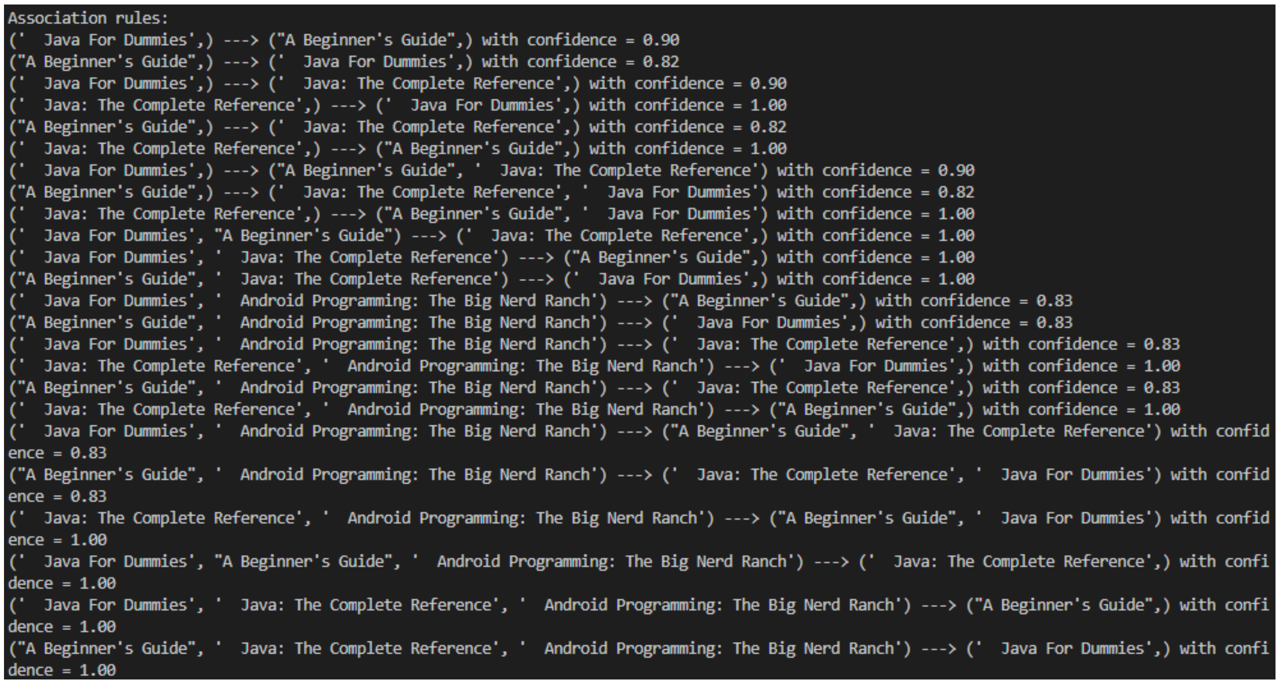


```
Time taken by each algorithm:

      Algorithm   Runtime
2      FPGrowth  0.021995
1       Apriori  0.031999
0  BruteApriori  0.062037
```

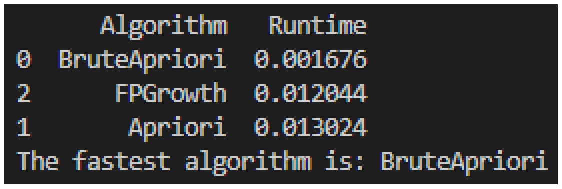
2. Support: 0.2 (20%), Confidence: 0.7 (70%)



```
Brute-Forced Apriori:

Frequent patterns:

('  Head First Java 2nd Edition',), support: 0.20
('  Java 8 Pocket Guide',), support: 0.20
('  Java: The Complete Reference',), support: 0.45
("A Beginner's Guide",), support: 0.55
('  Beginning Programming with Java',), support: 0.25
('  Java For Dummies',), support: 0.50
('  Android Programming: The Big Nerd Ranch',), support: 0.50
('  Java: The Complete Reference', "A Beginner's Guide"), support: 0.45
('  Java: The Complete Reference', '  Java For Dummies'), support: 0.45
('  Java: The Complete Reference', '  Android Programming: The Big Nerd Ranch'), support: 0.25
("A Beginner's Guide", '  Java For Dummies'), support: 0.45
("A Beginner's Guide", '  Android Programming: The Big Nerd Ranch'), support: 0.30
('  Java For Dummies', '  Android Programming: The Big Nerd Ranch'), support: 0.30
('  Java: The Complete Reference', "A Beginner's Guide", '  Java For Dummies'), support: 0.45
('  Java: The Complete Reference', "A Beginner's Guide", '  Android Programming: The Big Nerd Ranch'), support: 0.25
('  Java: The Complete Reference', '  Java For Dummies', '  Android Programming: The Big Nerd Ranch'), support: 0.25
("A Beginner's Guide", '  Java For Dummies', '  Android Programming: The Big Nerd Ranch'), support: 0.25
('  Java: The Complete Reference', "A Beginner's Guide", '  Java For Dummies', '  Android Programming: The Big Nerd Ranch'), support: 0.25
```

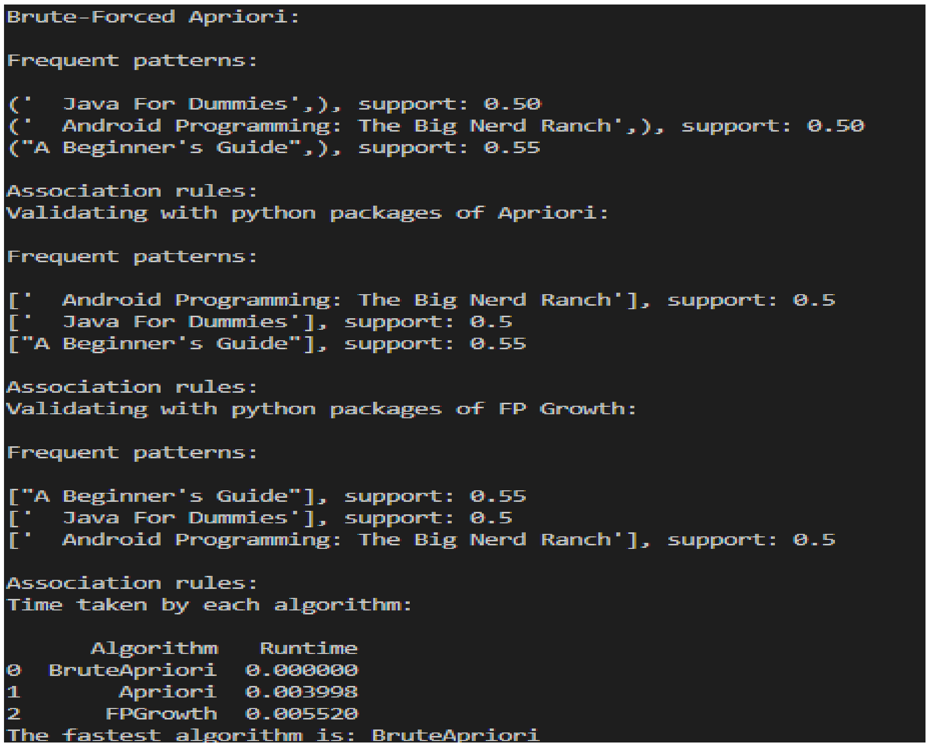


```
Association rules:
('  Java For Dummies',) ---> ("A Beginner's Guide",) with confidence = 0.90
("A Beginner's Guide",) ---> ('  Java For Dummies',) with confidence = 0.82
('  Java For Dummies',) ---> ('  Java: The Complete Reference',) with confidence = 0.90
('  Java: The Complete Reference',) ---> ('  Java For Dummies',) with confidence = 1.00
("A Beginner's Guide",) ---> ('  Java: The Complete Reference',) with confidence = 0.82
('  Java: The Complete Reference',) ---> ("A Beginner's Guide",) with confidence = 1.00
('  Java For Dummies',) ---> ("A Beginner's Guide", '  Java: The Complete Reference') with confidence = 0.90
("A Beginner's Guide",) ---> ('  Java: The Complete Reference', '  Java For Dummies') with confidence = 0.82
('  Java: The Complete Reference',) ---> ("A Beginner's Guide", '  Java For Dummies') with confidence = 1.00
('  Java For Dummies', "A Beginner's Guide") ---> ('  Java: The Complete Reference',) with confidence = 1.00
('  Java For Dummies', '  Java: The Complete Reference') ---> ("A Beginner's Guide",) with confidence = 1.00
("A Beginner's Guide", '  Java: The Complete Reference') ---> ('  Java For Dummies',) with confidence = 1.00
('  Java For Dummies', '  Android Programming: The Big Nerd Ranch') ---> ("A Beginner's Guide",) with confidence = 0.83
("A Beginner's Guide", '  Android Programming: The Big Nerd Ranch') ---> ('  Java For Dummies',) with confidence = 0.83
('  Java For Dummies', '  Android Programming: The Big Nerd Ranch') ---> ('  Java: The Complete Reference',) with confidence = 0.83
('  Java: The Complete Reference', '  Android Programming: The Big Nerd Ranch') ---> ('  Java For Dummies',) with confidence = 1.00
("A Beginner's Guide", '  Android Programming: The Big Nerd Ranch') ---> ('  Java: The Complete Reference',) with confidence = 0.83
('  Java: The Complete Reference', '  Android Programming: The Big Nerd Ranch') ---> ("A Beginner's Guide",) with confidence = 1.00
('  Java For Dummies', '  Android Programming: The Big Nerd Ranch') ---> ("A Beginner's Guide", '  Java: The Complete Reference') with confid
ence = 0.83
("A Beginner's Guide", '  Android Programming: The Big Nerd Ranch') ---> ('  Java: The Complete Reference', '  Java For Dummies') with confid
ence = 0.83
('  Java: The Complete Reference', '  Android Programming: The Big Nerd Ranch') ---> ("A Beginner's Guide", '  Java For Dummies') with confid
ence = 1.00
('  Java For Dummies', "A Beginner's Guide", '  Android Programming: The Big Nerd Ranch') ---> ('  Java: The Complete Reference',) with confi
dence = 1.00
('  Java For Dummies', '  Java: The Complete Reference', '  Android Programming: The Big Nerd Ranch') ---> ("A Beginner's Guide",) with confi
dence = 1.00
("A Beginner's Guide", '  Java: The Complete Reference', '  Android Programming: The Big Nerd Ranch') ---> ('  Java For Dummies',) with confi
dence = 1.00
```



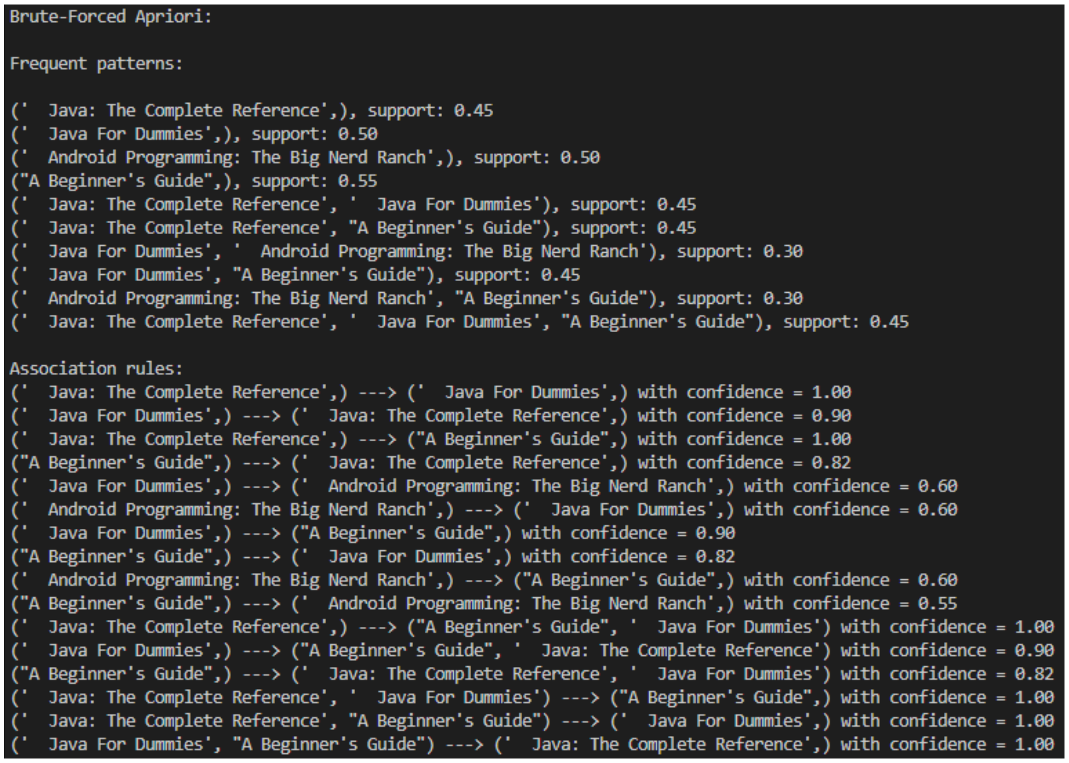
```
      Algorithm   Runtime
0  BruteApriori  0.001676
2      FPGrowth  0.012044
1       Apriori  0.013024
The fastest algorithm is: BruteApriori
```

3. Support: 0.5 (50%), Confidence: 0.7 (70%)



```
Brute-Forced Apriori:

Frequent patterns:

('  Java For Dummies',), support: 0.50
('  Android Programming: The Big Nerd Ranch',), support: 0.50
("A Beginner's Guide",), support: 0.55

Association rules:
Validating with python packages of Apriori:

Frequent patterns:

['  Android Programming: The Big Nerd Ranch'], support: 0.5
['  Java For Dummies'], support: 0.5
["A Beginner's Guide"], support: 0.55

Association rules:
Validating with python packages of FP Growth:

Frequent patterns:

["A Beginner's Guide"], support: 0.55
['  Java For Dummies'], support: 0.5
['  Android Programming: The Big Nerd Ranch'], support: 0.5

Association rules:
Time taken by each algorithm:

      Algorithm   Runtime
0  BruteApriori  0.000000
1       Apriori  0.003998
2      FPGrowth  0.005520
The fastest algorithm is: BruteApriori
```

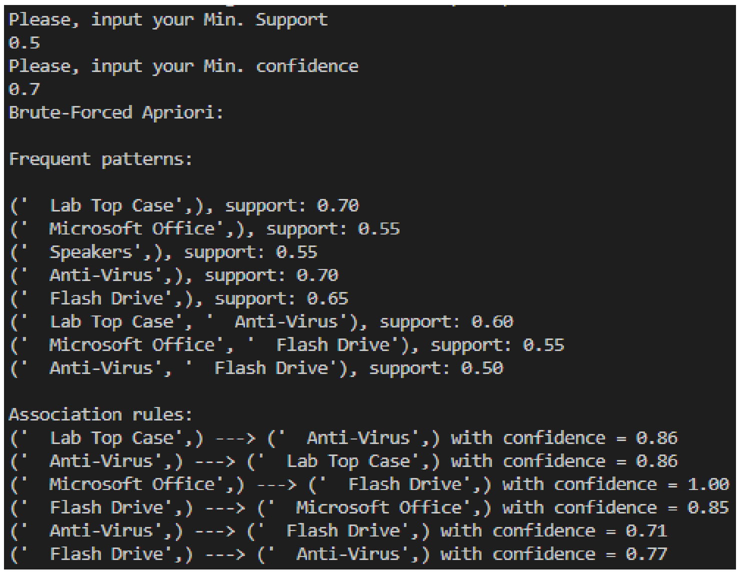
4. Support: 0.3 (30%), Confidence: 0.5 (50%)



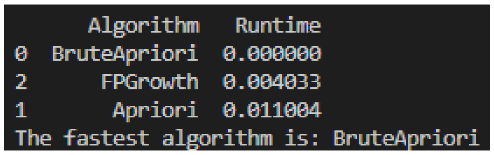
```
Brute-Forced Apriori:

Frequent patterns:

('  Java: The Complete Reference',), support: 0.45
('  Java For Dummies',), support: 0.50
('  Android Programming: The Big Nerd Ranch',), support: 0.50
("A Beginner's Guide",), support: 0.55
('  Java: The Complete Reference', '  Java For Dummies'), support: 0.45
('  Java: The Complete Reference', "A Beginner's Guide"), support: 0.45
('  Java For Dummies', '  Android Programming: The Big Nerd Ranch'), support: 0.30
('  Java For Dummies', "A Beginner's Guide"), support: 0.45
('  Android Programming: The Big Nerd Ranch', "A Beginner's Guide"), support: 0.30
('  Java: The Complete Reference', '  Java For Dummies', "A Beginner's Guide"), support: 0.45

Association rules:
('  Java: The Complete Reference',) ---> ('  Java For Dummies',) with confidence = 1.00
('  Java For Dummies',) ---> ('  Java: The Complete Reference',) with confidence = 0.90
('  Java: The Complete Reference',) ---> ("A Beginner's Guide",) with confidence = 1.00
("A Beginner's Guide",) ---> ('  Java: The Complete Reference',) with confidence = 0.82
('  Java For Dummies',) ---> ('  Android Programming: The Big Nerd Ranch',) with confidence = 0.60
('  Android Programming: The Big Nerd Ranch',) ---> ('  Java For Dummies',) with confidence = 0.60
('  Java For Dummies',) ---> ("A Beginner's Guide",) with confidence = 0.90
("A Beginner's Guide",) ---> ('  Java For Dummies',) with confidence = 0.82
('  Android Programming: The Big Nerd Ranch',) ---> ("A Beginner's Guide",) with confidence = 0.60
("A Beginner's Guide",) ---> ('  Android Programming: The Big Nerd Ranch',) with confidence = 0.55
('  Java: The Complete Reference',) ---> ("A Beginner's Guide", '  Java For Dummies') with confidence = 1.00
('  Java For Dummies',) ---> ("A Beginner's Guide", '  Java: The Complete Reference') with confidence = 0.90
("A Beginner's Guide",) ---> ('  Java: The Complete Reference', '  Java For Dummies') with confidence = 0.82
('  Java: The Complete Reference', '  Java For Dummies') ---> ("A Beginner's Guide",) with confidence = 1.00
('  Java: The Complete Reference', "A Beginner's Guide") ---> ('  Java For Dummies',) with confidence = 1.00
('  Java For Dummies', "A Beginner's Guide") ---> ('  Java: The Complete Reference',) with confidence = 1.00
```

For BestBuy Dataset:

5. Support: 0.5 (50%), Confidence: 0.7 (70%)

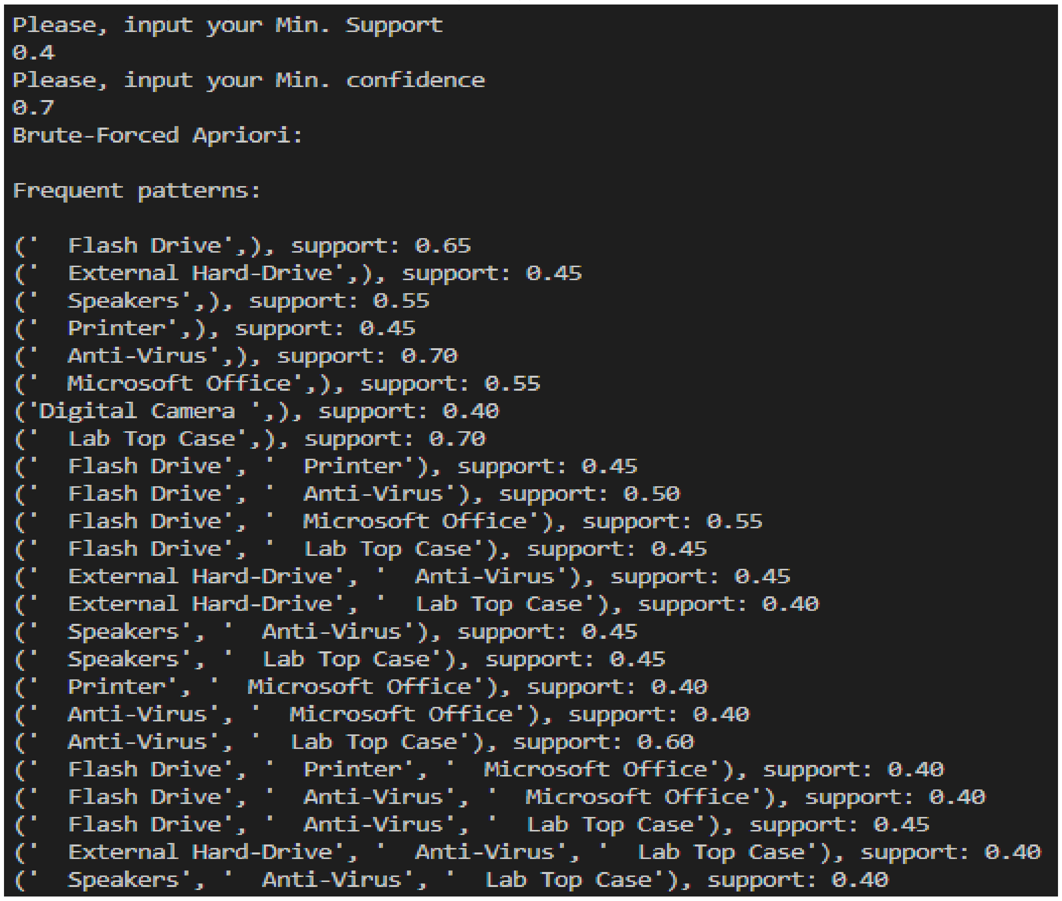


```
Please, input your Min. Support
0.5
Please, input your Min. confidence
0.7
Brute-Forced Apriori:

Frequent patterns:

('  Lab Top Case',), support: 0.70
('  Microsoft Office',), support: 0.55
('  Speakers',), support: 0.55
('  Anti-Virus',), support: 0.70
('  Flash Drive',), support: 0.65
('  Lab Top Case', '  Anti-Virus'), support: 0.60
('  Microsoft Office', '  Flash Drive'), support: 0.55
('  Anti-Virus', '  Flash Drive'), support: 0.50

Association rules:
('  Lab Top Case',) ---> ('  Anti-Virus',) with confidence = 0.86
('  Anti-Virus',) ---> ('  Lab Top Case',) with confidence = 0.86
('  Microsoft Office',) ---> ('  Flash Drive',) with confidence = 1.00
('  Flash Drive',) ---> ('  Microsoft Office',) with confidence = 0.85
('  Anti-Virus',) ---> ('  Flash Drive',) with confidence = 0.71
('  Flash Drive',) ---> ('  Anti-Virus',) with confidence = 0.77
```

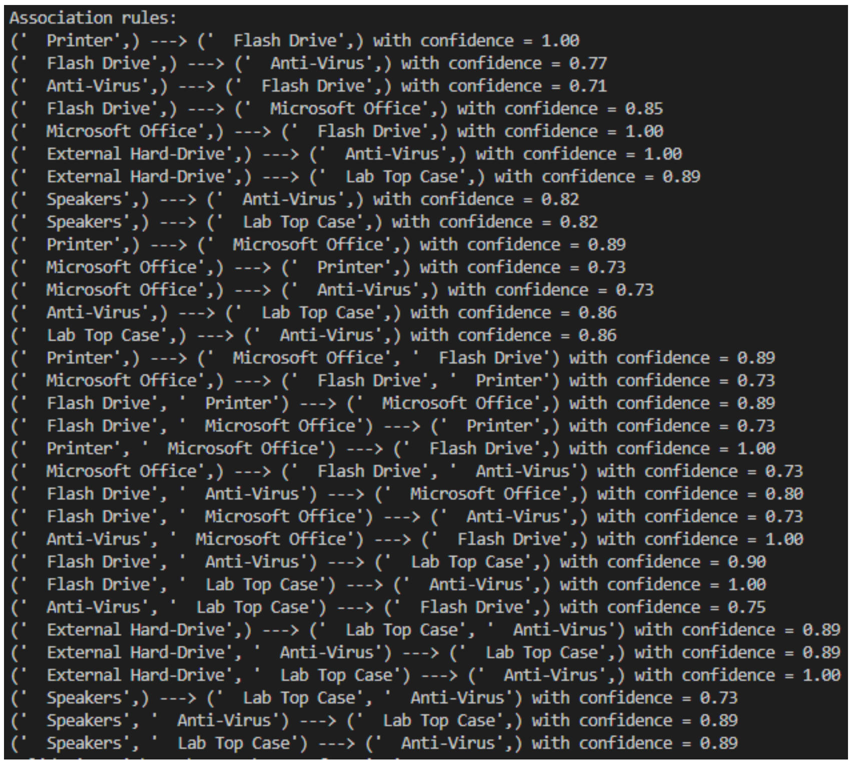


```
     Algorithm   Runtime
0  BruteApriori  0.000000
2      FPGrowth  0.004033
1       Apriori  0.011004
The fastest algorithm is: BruteApriori
```

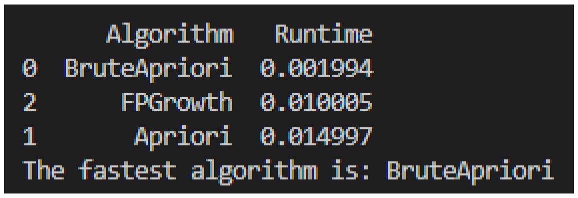
6. Support: 0.4 (40%), Confidence: 0.7 (70%)



```
Please, input your Min. Support
0.4
Please, input your Min. confidence
0.7
Brute-Forced Apriori:

Frequent patterns:

('  Flash Drive',), support: 0.65
('  External Hard-Drive',), support: 0.45
('  Speakers',), support: 0.55
('  Printer',), support: 0.45
('  Anti-Virus',), support: 0.70
('  Microsoft Office',), support: 0.55
('Digital Camera ',), support: 0.40
('  Lab Top Case',), support: 0.70
('  Flash Drive', '  Printer'), support: 0.45
('  Flash Drive', '  Anti-Virus'), support: 0.50
('  Flash Drive', '  Microsoft Office'), support: 0.55
('  Flash Drive', '  Lab Top Case'), support: 0.45
('  External Hard-Drive', '  Anti-Virus'), support: 0.45
('  External Hard-Drive', '  Lab Top Case'), support: 0.40
('  Speakers', '  Anti-Virus'), support: 0.45
('  Speakers', '  Lab Top Case'), support: 0.45
('  Printer', '  Microsoft Office'), support: 0.40
('  Anti-Virus', '  Microsoft Office'), support: 0.40
('  Anti-Virus', '  Lab Top Case'), support: 0.60
('  Flash Drive', '  Printer', '  Microsoft Office'), support: 0.40
('  Flash Drive', '  Anti-Virus', '  Microsoft Office'), support: 0.40
('  Flash Drive', '  Anti-Virus', '  Lab Top Case'), support: 0.45
('  External Hard-Drive', '  Anti-Virus', '  Lab Top Case'), support: 0.40
('  Speakers', '  Anti-Virus', '  Lab Top Case'), support: 0.40
```

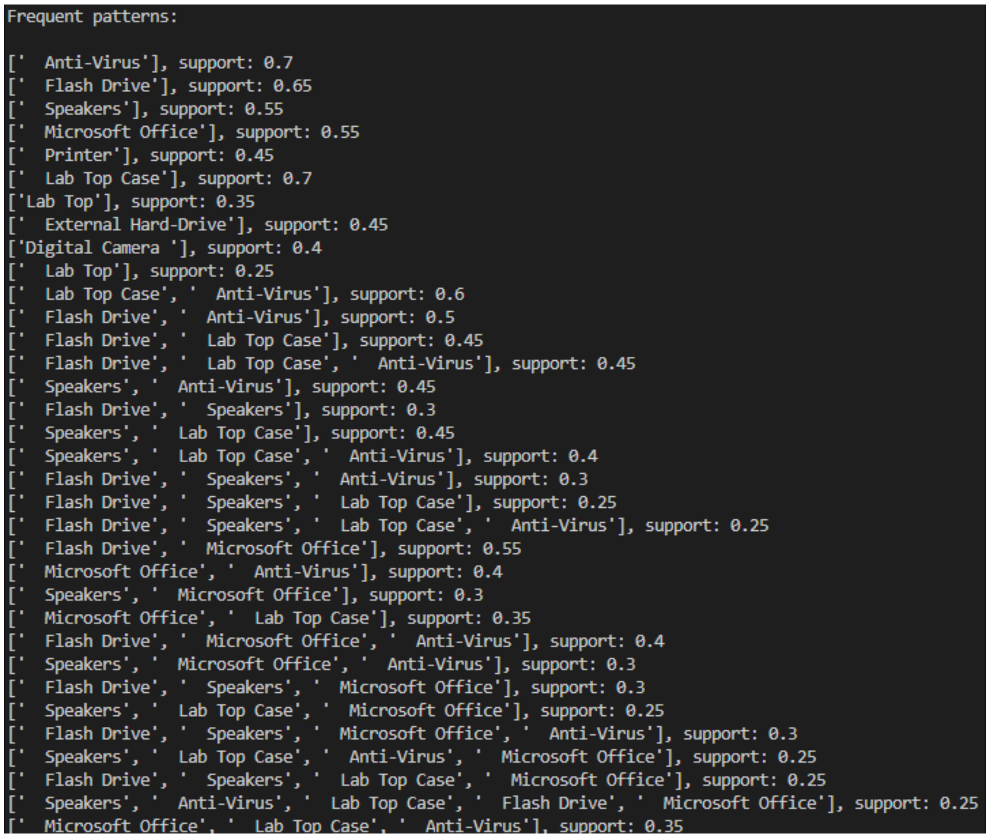


```
Association rules:
('  Printer',) ---> ('  Flash Drive',) with confidence = 1.00
('  Flash Drive',) ---> ('  Anti-Virus',) with confidence = 0.77
('  Anti-Virus',) ---> ('  Flash Drive',) with confidence = 0.71
('  Flash Drive',) ---> ('  Microsoft Office',) with confidence = 0.85
('  Microsoft Office',) ---> ('  Flash Drive',) with confidence = 1.00
('  External Hard-Drive',) ---> ('  Anti-Virus',) with confidence = 1.00
('  External Hard-Drive',) ---> ('  Lab Top Case',) with confidence = 0.89
('  Speakers',) ---> ('  Anti-Virus',) with confidence = 0.82
('  Speakers',) ---> ('  Lab Top Case',) with confidence = 0.82
('  Printer',) ---> ('  Microsoft Office',) with confidence = 0.89
('  Microsoft Office',) ---> ('  Printer',) with confidence = 0.73
('  Microsoft Office',) ---> ('  Anti-Virus',) with confidence = 0.73
('  Anti-Virus',) ---> ('  Lab Top Case',) with confidence = 0.86
('  Lab Top Case',) ---> ('  Anti-Virus',) with confidence = 0.86
('  Printer',) ---> ('  Microsoft Office', '  Flash Drive') with confidence = 0.89
('  Microsoft Office',) ---> ('  Flash Drive', '  Printer') with confidence = 0.73
('  Flash Drive', '  Printer') ---> ('  Microsoft Office',) with confidence = 0.89
('  Flash Drive', '  Microsoft Office') ---> ('  Printer',) with confidence = 0.73
('  Printer', '  Microsoft Office') ---> ('  Flash Drive',) with confidence = 1.00
('  Microsoft Office',) ---> ('  Flash Drive', '  Anti-Virus') with confidence = 0.73
('  Flash Drive', '  Anti-Virus') ---> ('  Microsoft Office',) with confidence = 0.80
('  Flash Drive', '  Microsoft Office') ---> ('  Anti-Virus',) with confidence = 0.73
('  Anti-Virus', '  Microsoft Office') ---> ('  Flash Drive',) with confidence = 1.00
('  Flash Drive', '  Anti-Virus') ---> ('  Lab Top Case',) with confidence = 0.90
('  Flash Drive', '  Lab Top Case') ---> ('  Anti-Virus',) with confidence = 1.00
('  Anti-Virus', '  Lab Top Case') ---> ('  Flash Drive',) with confidence = 0.75
('  External Hard-Drive',) ---> ('  Lab Top Case', '  Anti-Virus') with confidence = 0.89
('  External Hard-Drive', '  Anti-Virus') ---> ('  Lab Top Case',) with confidence = 0.89
('  External Hard-Drive', '  Lab Top Case') ---> ('  Anti-Virus',) with confidence = 1.00
('  Speakers',) ---> ('  Lab Top Case', '  Anti-Virus') with confidence = 0.73
('  Speakers', '  Anti-Virus') ---> ('  Lab Top Case',) with confidence = 0.89
('  Speakers', '  Lab Top Case') ---> ('  Anti-Virus',) with confidence = 0.89
```

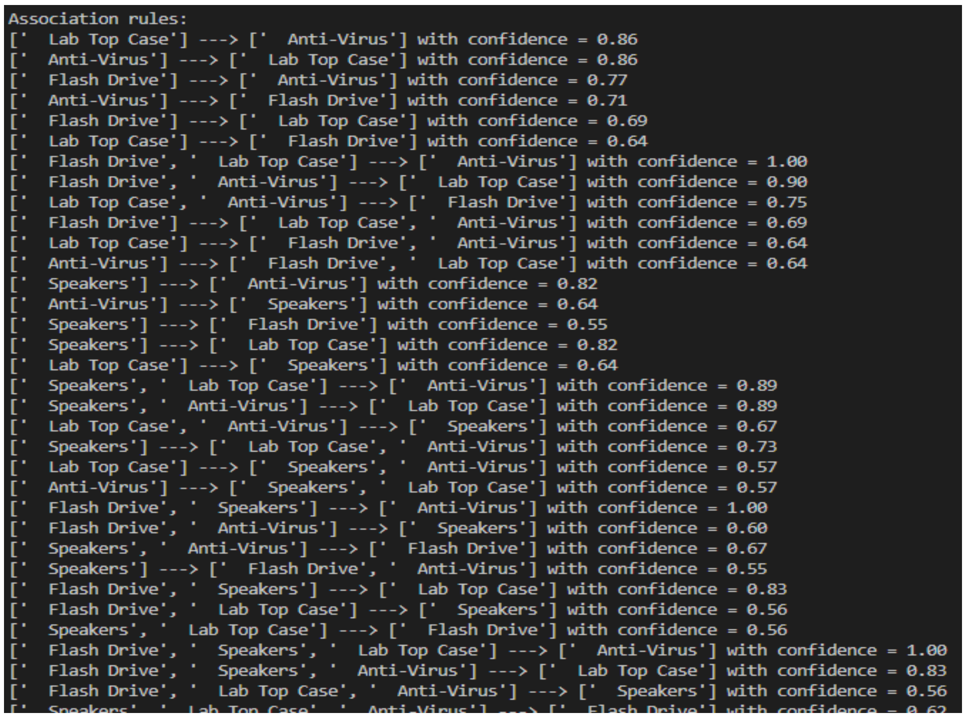


```
      Algorithm   Runtime
0  BruteApriori  0.001994
2      FPGrowth  0.010005
1       Apriori  0.014997
The fastest algorithm is: BruteApriori
```

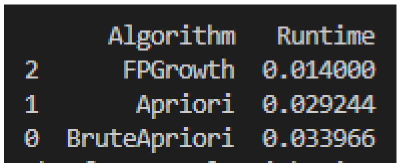
7. Support: 0.01 (1%), Confidence: 0.7 (70%)



```
Frequent patterns:

['  Anti-Virus'], support: 0.7
['  Flash Drive'], support: 0.65
['  Speakers'], support: 0.55
['  Microsoft Office'], support: 0.55
['  Printer'], support: 0.45
['  Lab Top Case'], support: 0.7
['Lab Top'], support: 0.35
['  External Hard-Drive'], support: 0.45
['Digital Camera '], support: 0.4
['  Lab Top'], support: 0.25
['  Lab Top Case', '  Anti-Virus'], support: 0.6
['  Flash Drive', '  Anti-Virus'], support: 0.5
['  Flash Drive', '  Lab Top Case'], support: 0.45
['  Flash Drive', '  Lab Top Case', '  Anti-Virus'], support: 0.45
['  Speakers', '  Anti-Virus'], support: 0.45
['  Flash Drive', '  Speakers'], support: 0.3
['  Speakers', '  Lab Top Case'], support: 0.45
['  Speakers', '  Lab Top Case', '  Anti-Virus'], support: 0.4
['  Flash Drive', '  Speakers', '  Anti-Virus'], support: 0.3
['  Flash Drive', '  Speakers', '  Lab Top Case'], support: 0.25
['  Flash Drive', '  Speakers', '  Lab Top Case', '  Anti-Virus'], support: 0.25
['  Flash Drive', '  Microsoft Office'], support: 0.55
['  Microsoft Office', '  Anti-Virus'], support: 0.4
['  Speakers', '  Microsoft Office'], support: 0.3
['  Microsoft Office', '  Lab Top Case'], support: 0.35
['  Flash Drive', '  Microsoft Office', '  Anti-Virus'], support: 0.4
['  Speakers', '  Microsoft Office', '  Anti-Virus'], support: 0.3
['  Flash Drive', '  Speakers', '  Microsoft Office'], support: 0.3
['  Speakers', '  Lab Top Case', '  Microsoft Office'], support: 0.25
['  Flash Drive', '  Speakers', '  Microsoft Office', '  Anti-Virus'], support: 0.3
['  Speakers', '  Lab Top Case', '  Anti-Virus', '  Microsoft Office'], support: 0.25
['  Flash Drive', '  Speakers', '  Lab Top Case', '  Microsoft Office'], support: 0.25
['  Speakers', '  Anti-Virus', '  Lab Top Case', '  Flash Drive', '  Microsoft Office'], support: 0.25
['  Microsoft Office', '  Lab Top Case', '  Anti-Virus'], support: 0.35
```



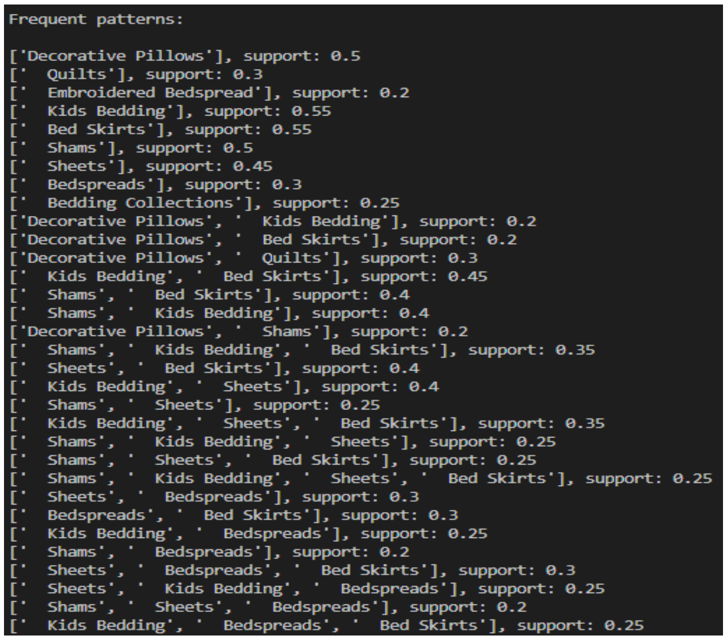
```
Association rules:
['  Lab Top Case'] ---> ['  Anti-Virus'] with confidence = 0.86
['  Anti-Virus'] ---> ['  Lab Top Case'] with confidence = 0.86
['  Flash Drive'] ---> ['  Anti-Virus'] with confidence = 0.77
['  Anti-Virus'] ---> ['  Flash Drive'] with confidence = 0.71
['  Flash Drive'] ---> ['  Lab Top Case'] with confidence = 0.69
['  Lab Top Case'] ---> ['  Flash Drive'] with confidence = 0.64
['  Flash Drive', '  Lab Top Case'] ---> ['  Anti-Virus'] with confidence = 1.00
['  Flash Drive', '  Anti-Virus'] ---> ['  Lab Top Case'] with confidence = 0.90
['  Lab Top Case', '  Anti-Virus'] ---> ['  Flash Drive'] with confidence = 0.75
['  Flash Drive'] ---> ['  Lab Top Case', '  Anti-Virus'] with confidence = 0.69
['  Lab Top Case'] ---> ['  Flash Drive', '  Anti-Virus'] with confidence = 0.64
['  Anti-Virus'] ---> ['  Flash Drive', '  Lab Top Case'] with confidence = 0.64
['  Speakers'] ---> ['  Anti-Virus'] with confidence = 0.82
['  Anti-Virus'] ---> ['  Speakers'] with confidence = 0.64
['  Speakers'] ---> ['  Flash Drive'] with confidence = 0.55
['  Speakers'] ---> ['  Lab Top Case'] with confidence = 0.82
['  Lab Top Case'] ---> ['  Speakers'] with confidence = 0.64
['  Speakers', '  Lab Top Case'] ---> ['  Anti-Virus'] with confidence = 0.89
['  Speakers', '  Anti-Virus'] ---> ['  Lab Top Case'] with confidence = 0.89
['  Lab Top Case', '  Anti-Virus'] ---> ['  Speakers'] with confidence = 0.67
['  Speakers'] ---> ['  Lab Top Case', '  Anti-Virus'] with confidence = 0.73
['  Lab Top Case'] ---> ['  Speakers', '  Anti-Virus'] with confidence = 0.57
['  Anti-Virus'] ---> ['  Speakers', '  Lab Top Case'] with confidence = 0.57
['  Flash Drive', '  Speakers'] ---> ['  Anti-Virus'] with confidence = 1.00
['  Flash Drive', '  Anti-Virus'] ---> ['  Speakers'] with confidence = 0.60
['  Speakers', '  Anti-Virus'] ---> ['  Flash Drive'] with confidence = 0.67
['  Speakers'] ---> ['  Flash Drive', '  Anti-Virus'] with confidence = 0.55
['  Flash Drive', '  Speakers'] ---> ['  Lab Top Case'] with confidence = 0.83
['  Flash Drive', '  Lab Top Case'] ---> ['  Speakers'] with confidence = 0.56
['  Speakers', '  Lab Top Case'] ---> ['  Flash Drive'] with confidence = 0.56
['  Flash Drive', '  Speakers', '  Lab Top Case'] ---> ['  Anti-Virus'] with confidence = 1.00
['  Flash Drive', '  Speakers', '  Anti-Virus'] ---> ['  Lab Top Case'] with confidence = 0.83
['  Flash Drive', '  Lab Top Case', '  Anti-Virus'] ---> ['  Speakers'] with confidence = 0.56
['  Speakers', '  Lab Top Case', '  Anti-Virus'] ---> ['  Flash Drive'] with confidence = 0.62
```



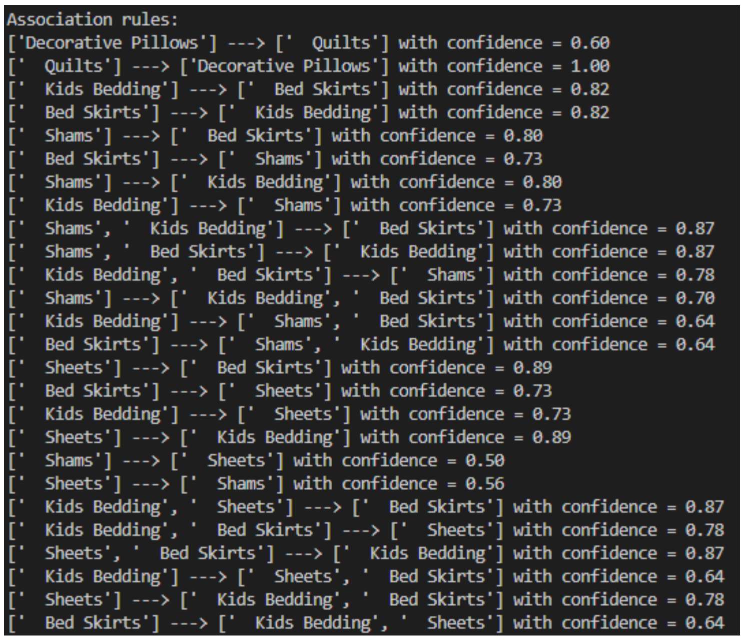
```
      Algorithm   Runtime
2      FPGrowth  0.014000
1       Apriori  0.029244
0  BruteApriori  0.033966
```

For Kmart Dataset:

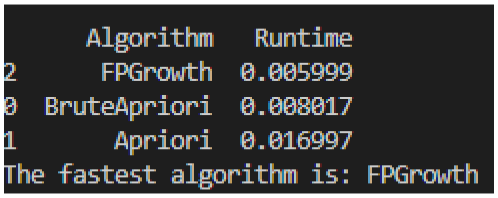
8. Support: 0.2 (20%), Confidence: 0.5 (50%)



```
Frequent patterns:

['Decorative Pillows'], support: 0.5
['  Quilts'], support: 0.3
['  Embroidered Bedspread'], support: 0.2
['  Kids Bedding'], support: 0.55
['  Bed Skirts'], support: 0.55
['  Shams'], support: 0.5
['  Sheets'], support: 0.45
['  Bedspreads'], support: 0.3
['  Bedding Collections'], support: 0.25
['Decorative Pillows', '  Kids Bedding'], support: 0.2
['Decorative Pillows', '  Bed Skirts'], support: 0.2
['Decorative Pillows', '  Quilts'], support: 0.3
['  Kids Bedding', '  Bed Skirts'], support: 0.45
['  Shams', '  Bed Skirts'], support: 0.4
['  Shams', '  Kids Bedding'], support: 0.4
['Decorative Pillows', '  Shams'], support: 0.2
['  Shams', '  Kids Bedding', '  Bed Skirts'], support: 0.35
['  Sheets', '  Bed Skirts'], support: 0.4
['  Kids Bedding', '  Sheets'], support: 0.4
['  Shams', '  Sheets'], support: 0.25
['  Kids Bedding', '  Sheets', '  Bed Skirts'], support: 0.35
['  Shams', '  Kids Bedding', '  Sheets'], support: 0.25
['  Shams', '  Sheets', '  Bed Skirts'], support: 0.25
['  Shams', '  Kids Bedding', '  Sheets', '  Bed Skirts'], support: 0.25
['  Sheets', '  Bedspreads'], support: 0.3
['  Bedspreads', '  Bed Skirts'], support: 0.3
['  Kids Bedding', '  Bedspreads'], support: 0.25
['  Shams', '  Bedspreads'], support: 0.2
['  Sheets', '  Bedspreads', '  Bed Skirts'], support: 0.3
['  Sheets', '  Kids Bedding', '  Bedspreads'], support: 0.25
['  Shams', '  Sheets', '  Bedspreads'], support: 0.2
['  Kids Bedding', '  Bedspreads', '  Bed Skirts'], support: 0.25
```

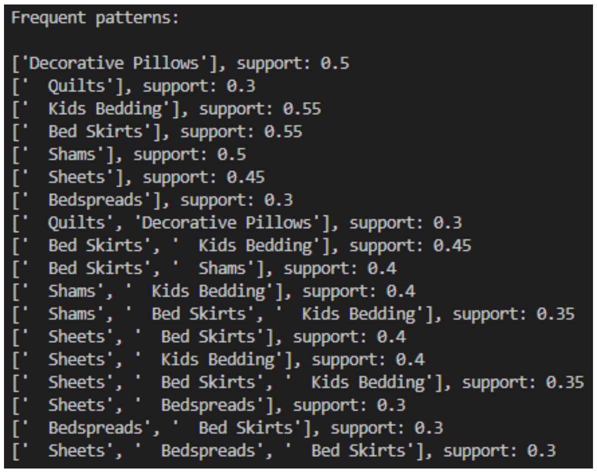


```
Association rules:
['Decorative Pillows'] ---> ['  Quilts'] with confidence = 0.60
['  Quilts'] ---> ['Decorative Pillows'] with confidence = 1.00
['  Kids Bedding'] ---> ['  Bed Skirts'] with confidence = 0.82
['  Bed Skirts'] ---> ['  Kids Bedding'] with confidence = 0.82
['  Shams'] ---> ['  Bed Skirts'] with confidence = 0.80
['  Bed Skirts'] ---> ['  Shams'] with confidence = 0.73
['  Shams'] ---> ['  Kids Bedding'] with confidence = 0.80
['  Kids Bedding'] ---> ['  Shams'] with confidence = 0.73
['  Shams', '  Kids Bedding'] ---> ['  Bed Skirts'] with confidence = 0.87
['  Shams', '  Bed Skirts'] ---> ['  Kids Bedding'] with confidence = 0.87
['  Kids Bedding', '  Bed Skirts'] ---> ['  Shams'] with confidence = 0.78
['  Shams'] ---> ['  Kids Bedding', '  Bed Skirts'] with confidence = 0.70
['  Kids Bedding'] ---> ['  Shams', '  Bed Skirts'] with confidence = 0.64
['  Bed Skirts'] ---> ['  Shams', '  Kids Bedding'] with confidence = 0.64
['  Sheets'] ---> ['  Bed Skirts'] with confidence = 0.89
['  Bed Skirts'] ---> ['  Sheets'] with confidence = 0.73
['  Kids Bedding'] ---> ['  Sheets'] with confidence = 0.73
['  Sheets'] ---> ['  Kids Bedding'] with confidence = 0.89
['  Shams'] ---> ['  Sheets'] with confidence = 0.50
['  Sheets'] ---> ['  Shams'] with confidence = 0.56
['  Kids Bedding', '  Sheets'] ---> ['  Bed Skirts'] with confidence = 0.87
['  Kids Bedding', '  Bed Skirts'] ---> ['  Sheets'] with confidence = 0.78
['  Sheets', '  Bed Skirts'] ---> ['  Kids Bedding'] with confidence = 0.87
['  Kids Bedding'] ---> ['  Sheets', '  Bed Skirts'] with confidence = 0.64
['  Sheets'] ---> ['  Kids Bedding', '  Bed Skirts'] with confidence = 0.78
['  Bed Skirts'] ---> ['  Kids Bedding', '  Sheets'] with confidence = 0.64
```

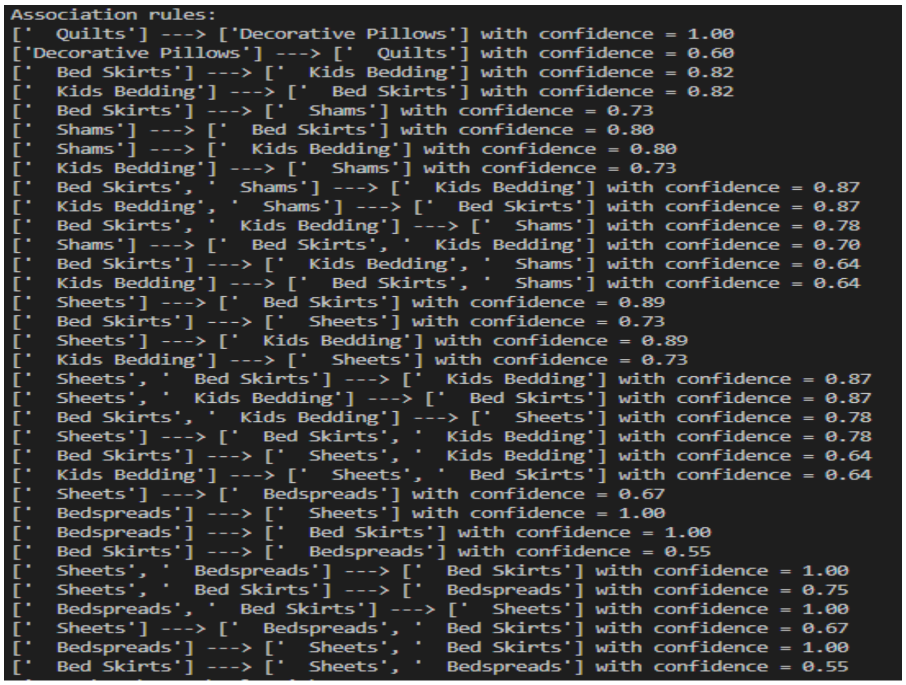


```
      Algorithm   Runtime
2      FPGrowth  0.005999
0  BruteApriori  0.008017
1       Apriori  0.016997
The fastest algorithm is: FPGrowth
```

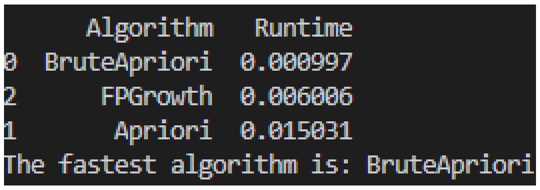
9. Support: 0.1 (10%), Confidence: 0.5 (50%)



```
Frequent patterns:

['Decorative Pillows'], support: 0.5
['  Quilts'], support: 0.3
['  Kids Bedding'], support: 0.55
['  Bed Skirts'], support: 0.55
['  Shams'], support: 0.5
['  Sheets'], support: 0.45
['  Bedspreads'], support: 0.3
['  Quilts', 'Decorative Pillows'], support: 0.3
['  Bed Skirts', '  Kids Bedding'], support: 0.45
['  Bed Skirts', '  Shams'], support: 0.4
['  Shams', '  Kids Bedding'], support: 0.4
['  Shams', '  Bed Skirts', '  Kids Bedding'], support: 0.35
['  Sheets', '  Bed Skirts'], support: 0.4
['  Sheets', '  Kids Bedding'], support: 0.4
['  Sheets', '  Bed Skirts', '  Kids Bedding'], support: 0.35
['  Sheets', '  Bedspreads'], support: 0.3
['  Bedspreads', '  Bed Skirts'], support: 0.3
['  Sheets', '  Bedspreads', '  Bed Skirts'], support: 0.3
```



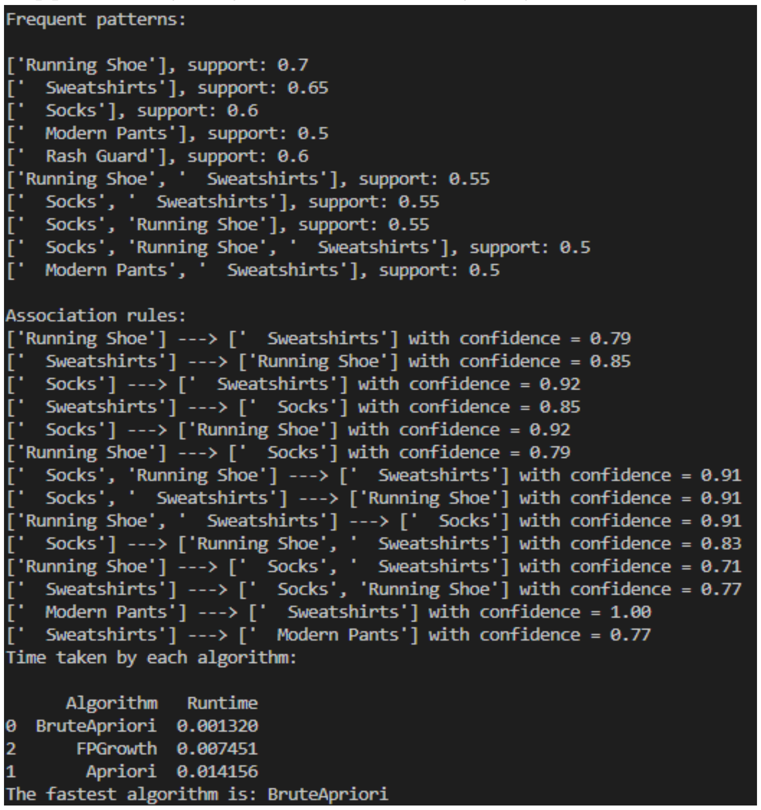
```
Association rules:
['  Quilts'] ---> ['Decorative Pillows'] with confidence = 1.00
['Decorative Pillows'] ---> ['  Quilts'] with confidence = 0.60
['  Bed Skirts'] ---> ['  Kids Bedding'] with confidence = 0.82
['  Kids Bedding'] ---> ['  Bed Skirts'] with confidence = 0.82
['  Bed Skirts'] ---> ['  Shams'] with confidence = 0.73
['  Shams'] ---> ['  Bed Skirts'] with confidence = 0.80
['  Shams'] ---> ['  Kids Bedding'] with confidence = 0.80
['  Kids Bedding'] ---> ['  Shams'] with confidence = 0.73
['  Bed Skirts', '  Shams'] ---> ['  Kids Bedding'] with confidence = 0.87
['  Kids Bedding', '  Shams'] ---> ['  Bed Skirts'] with confidence = 0.87
['  Bed Skirts', '  Kids Bedding'] ---> ['  Shams'] with confidence = 0.78
['  Shams'] ---> ['  Bed Skirts', '  Kids Bedding'] with confidence = 0.70
['  Bed Skirts'] ---> ['  Kids Bedding', '  Shams'] with confidence = 0.64
['  Kids Bedding'] ---> ['  Bed Skirts', '  Shams'] with confidence = 0.64
['  Sheets'] ---> ['  Bed Skirts'] with confidence = 0.89
['  Bed Skirts'] ---> ['  Sheets'] with confidence = 0.73
['  Sheets'] ---> ['  Kids Bedding'] with confidence = 0.89
['  Kids Bedding'] ---> ['  Sheets'] with confidence = 0.73
['  Sheets', '  Bed Skirts'] ---> ['  Kids Bedding'] with confidence = 0.87
['  Sheets', '  Kids Bedding'] ---> ['  Bed Skirts'] with confidence = 0.87
['  Bed Skirts', '  Kids Bedding'] ---> ['  Sheets'] with confidence = 0.78
['  Sheets'] ---> ['  Bed Skirts', '  Kids Bedding'] with confidence = 0.78
['  Bed Skirts'] ---> ['  Sheets', '  Kids Bedding'] with confidence = 0.64
['  Kids Bedding'] ---> ['  Sheets', '  Bed Skirts'] with confidence = 0.64
['  Sheets'] ---> ['  Bedspreads'] with confidence = 0.67
['  Bedspreads'] ---> ['  Sheets'] with confidence = 1.00
['  Bedspreads'] ---> ['  Bed Skirts'] with confidence = 1.00
['  Bed Skirts'] ---> ['  Bedspreads'] with confidence = 0.55
['  Sheets', '  Bedspreads'] ---> ['  Bed Skirts'] with confidence = 1.00
['  Sheets', '  Bed Skirts'] ---> ['  Bedspreads'] with confidence = 0.75
['  Bedspreads', '  Bed Skirts'] ---> ['  Sheets'] with confidence = 1.00
['  Sheets'] ---> ['  Bedspreads', '  Bed Skirts'] with confidence = 0.67
['  Bedspreads'] ---> ['  Sheets', '  Bed Skirts'] with confidence = 1.00
['  Bed Skirts'] ---> ['  Sheets', '  Bedspreads'] with confidence = 0.55
```



```
      Algorithm   Runtime
0  BruteApriori  0.000997
2      FPGrowth  0.006006
1       Apriori  0.015031
The fastest algorithm is: BruteApriori
```

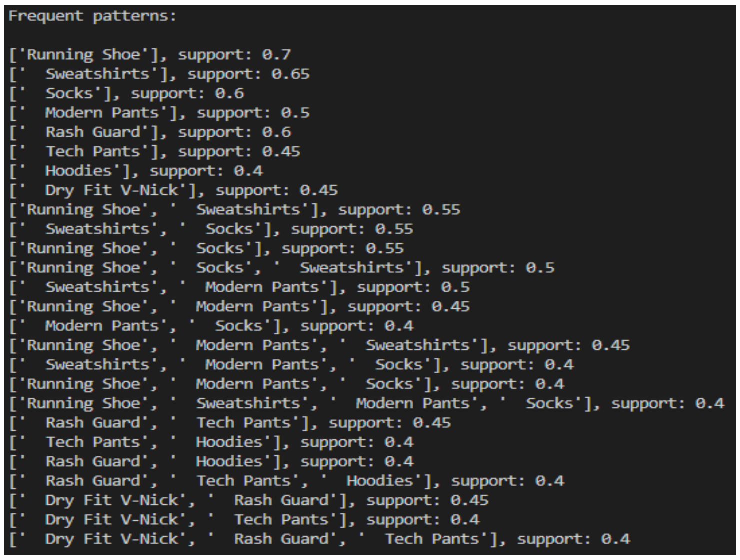
For Nike Dataset:

10. Support: 0.5 (50%), Confidence: 0.5 (50%)

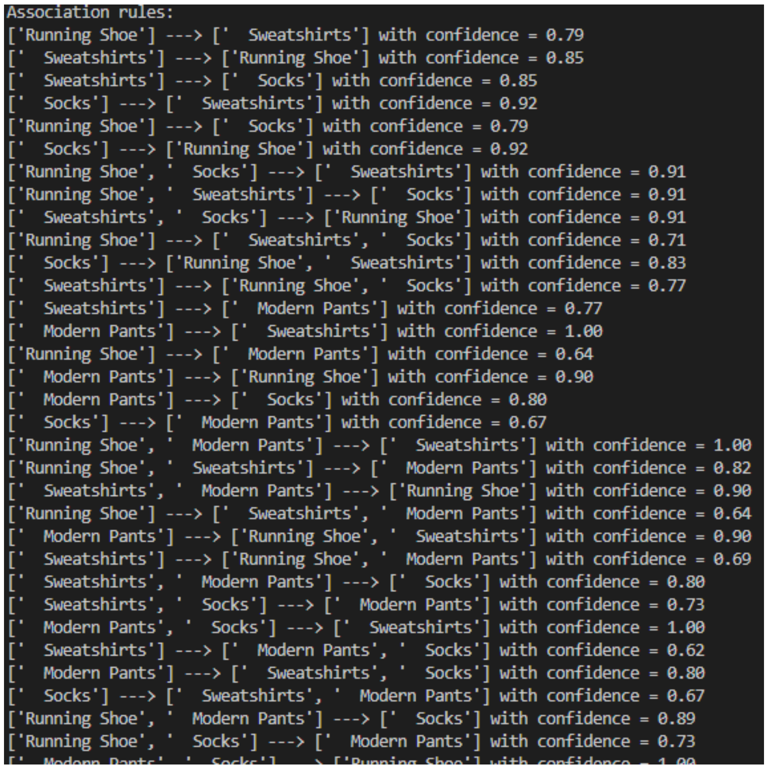


```
Frequent patterns:

['Running Shoe'], support: 0.7
['  Sweatshirts'], support: 0.65
['  Socks'], support: 0.6
['  Modern Pants'], support: 0.5
['  Rash Guard'], support: 0.6
['Running Shoe', '  Sweatshirts'], support: 0.55
['  Socks', '  Sweatshirts'], support: 0.55
['  Socks', 'Running Shoe'], support: 0.55
['  Socks', 'Running Shoe', '  Sweatshirts'], support: 0.5
['  Modern Pants', '  Sweatshirts'], support: 0.5

Association rules:
['Running Shoe'] ---> ['  Sweatshirts'] with confidence = 0.79
['  Sweatshirts'] ---> ['Running Shoe'] with confidence = 0.85
['  Socks'] ---> ['  Sweatshirts'] with confidence = 0.92
['  Sweatshirts'] ---> ['  Socks'] with confidence = 0.85
['  Socks'] ---> ['Running Shoe'] with confidence = 0.92
['Running Shoe'] ---> ['  Socks'] with confidence = 0.79
['  Socks', 'Running Shoe'] ---> ['  Sweatshirts'] with confidence = 0.91
['  Socks', '  Sweatshirts'] ---> ['Running Shoe'] with confidence = 0.91
['Running Shoe', '  Sweatshirts'] ---> ['  Socks'] with confidence = 0.91
['  Socks'] ---> ['Running Shoe', '  Sweatshirts'] with confidence = 0.83
['Running Shoe'] ---> ['  Socks', '  Sweatshirts'] with confidence = 0.71
['  Sweatshirts'] ---> ['  Socks', 'Running Shoe'] with confidence = 0.77
['  Modern Pants'] ---> ['  Sweatshirts'] with confidence = 1.00
['  Sweatshirts'] ---> ['  Modern Pants'] with confidence = 0.77
Time taken by each algorithm:

      Algorithm   Runtime
0  BruteApriori  0.001320
2      FPGrowth  0.007451
1       Apriori  0.014156
The fastest algorithm is: BruteApriori
```

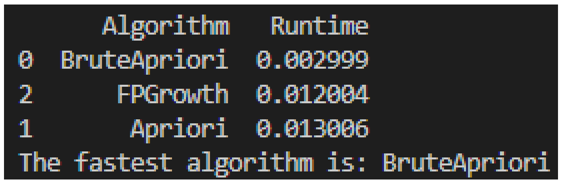
11. Support: 0.4 (40%), Confidence: 0.5 (50%)



```
Frequent patterns:

['Running Shoe'], support: 0.7
['  Sweatshirts'], support: 0.65
['  Socks'], support: 0.6
['  Modern Pants'], support: 0.5
['  Rash Guard'], support: 0.6
['  Tech Pants'], support: 0.45
['  Hoodies'], support: 0.4
['  Dry Fit V-Nick'], support: 0.45
['Running Shoe', '  Sweatshirts'], support: 0.55
['  Sweatshirts', '  Socks'], support: 0.55
['Running Shoe', '  Socks'], support: 0.55
['Running Shoe', '  Socks', '  Sweatshirts'], support: 0.5
['  Sweatshirts', '  Modern Pants'], support: 0.5
['Running Shoe', '  Modern Pants'], support: 0.45
['  Modern Pants', '  Socks'], support: 0.4
['Running Shoe', '  Modern Pants', '  Sweatshirts'], support: 0.45
['  Sweatshirts', '  Modern Pants', '  Socks'], support: 0.4
['Running Shoe', '  Modern Pants', '  Socks'], support: 0.4
['Running Shoe', '  Sweatshirts', '  Modern Pants', '  Socks'], support: 0.4
['  Rash Guard', '  Tech Pants'], support: 0.45
['  Tech Pants', '  Hoodies'], support: 0.4
['  Rash Guard', '  Hoodies'], support: 0.4
['  Rash Guard', '  Tech Pants', '  Hoodies'], support: 0.4
['  Dry Fit V-Nick', '  Rash Guard'], support: 0.45
['  Dry Fit V-Nick', '  Tech Pants'], support: 0.4
['  Dry Fit V-Nick', '  Rash Guard', '  Tech Pants'], support: 0.4
```



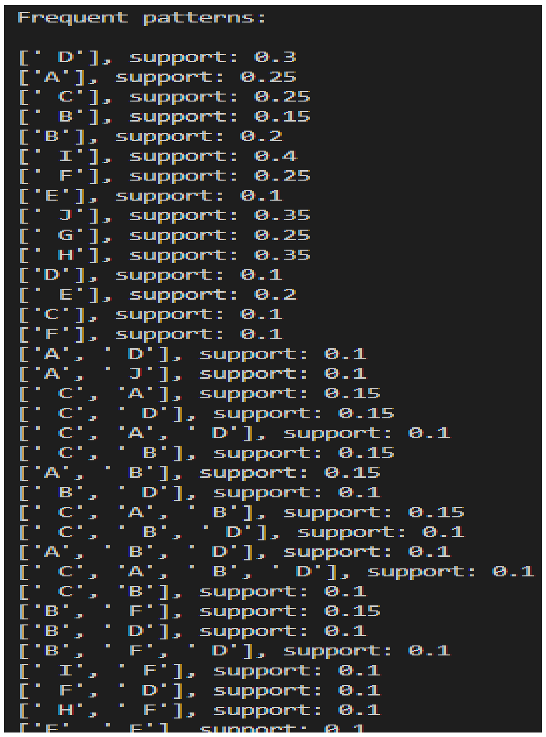
```
Association rules:
['Running Shoe'] ---> ['  Sweatshirts'] with confidence = 0.79
['  Sweatshirts'] ---> ['Running Shoe'] with confidence = 0.85
['  Sweatshirts'] ---> ['  Socks'] with confidence = 0.85
['  Socks'] ---> ['  Sweatshirts'] with confidence = 0.92
['Running Shoe'] ---> ['  Socks'] with confidence = 0.79
['  Socks'] ---> ['Running Shoe'] with confidence = 0.92
['Running Shoe', '  Socks'] ---> ['  Sweatshirts'] with confidence = 0.91
['Running Shoe', '  Sweatshirts'] ---> ['  Socks'] with confidence = 0.91
['  Sweatshirts', '  Socks'] ---> ['Running Shoe'] with confidence = 0.91
['Running Shoe'] ---> ['  Sweatshirts', '  Socks'] with confidence = 0.71
['  Socks'] ---> ['Running Shoe', '  Sweatshirts'] with confidence = 0.83
['  Sweatshirts'] ---> ['Running Shoe', '  Socks'] with confidence = 0.77
['  Sweatshirts'] ---> ['  Modern Pants'] with confidence = 0.77
['  Modern Pants'] ---> ['  Sweatshirts'] with confidence = 1.00
['Running Shoe'] ---> ['  Modern Pants'] with confidence = 0.64
['  Modern Pants'] ---> ['Running Shoe'] with confidence = 0.90
['  Modern Pants'] ---> ['  Socks'] with confidence = 0.80
['  Socks'] ---> ['  Modern Pants'] with confidence = 0.67
['Running Shoe', '  Modern Pants'] ---> ['  Sweatshirts'] with confidence = 1.00
['Running Shoe', '  Sweatshirts'] ---> ['  Modern Pants'] with confidence = 0.82
['  Sweatshirts', '  Modern Pants'] ---> ['Running Shoe'] with confidence = 0.90
['Running Shoe'] ---> ['  Sweatshirts', '  Modern Pants'] with confidence = 0.64
['  Modern Pants'] ---> ['Running Shoe', '  Sweatshirts'] with confidence = 0.90
['  Sweatshirts'] ---> ['Running Shoe', '  Modern Pants'] with confidence = 0.69
['  Sweatshirts', '  Modern Pants'] ---> ['  Socks'] with confidence = 0.80
['  Sweatshirts', '  Socks'] ---> ['  Modern Pants'] with confidence = 0.73
['  Modern Pants', '  Socks'] ---> ['  Sweatshirts'] with confidence = 1.00
['  Sweatshirts'] ---> ['  Modern Pants', '  Socks'] with confidence = 0.62
['  Modern Pants'] ---> ['  Sweatshirts', '  Socks'] with confidence = 0.80
['  Socks'] ---> ['  Sweatshirts', '  Modern Pants'] with confidence = 0.67
['Running Shoe', '  Modern Pants'] ---> ['  Socks'] with confidence = 0.89
['Running Shoe', '  Socks'] ---> ['  Modern Pants'] with confidence = 0.73
['  Modern Pants', '  Socks'] ---> ['Running Shoe'] with confidence = 1.00
```



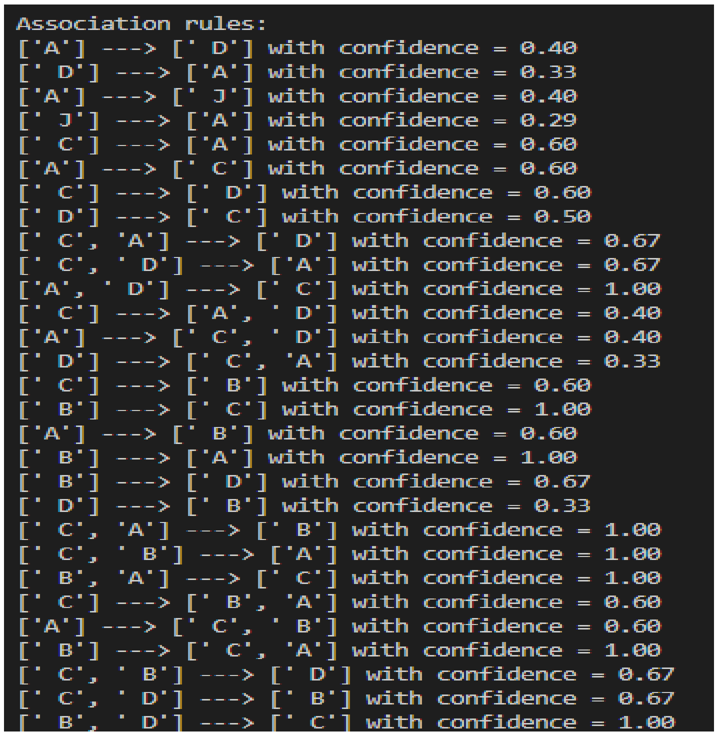
```
       Algorithm   Runtime
0  BruteApriori  0.002999
2      FPGrowth  0.012004
1       Apriori  0.013006
The fastest algorithm is: BruteApriori
```

For Generic Dataset:

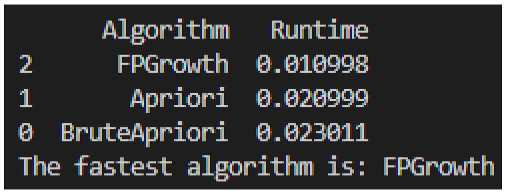
12. Support: 0.1 (10%), Confidence: 0.1 (10%)



```
Frequent patterns:

[' D'], support: 0.3
['A'], support: 0.25
[' C'], support: 0.25
[' B'], support: 0.15
['B'], support: 0.2
[' I'], support: 0.4
[' F'], support: 0.25
['E'], support: 0.1
[' J'], support: 0.35
[' G'], support: 0.25
[' H'], support: 0.35
['D'], support: 0.1
[' E'], support: 0.2
['C'], support: 0.1
['F'], support: 0.1
['A', ' D'], support: 0.1
['A', ' J'], support: 0.1
[' C', 'A'], support: 0.15
[' C', ' D'], support: 0.15
[' C', 'A', ' D'], support: 0.1
[' C', ' B'], support: 0.15
['A', ' B'], support: 0.15
[' B', ' D'], support: 0.1
[' C', 'A', ' B'], support: 0.15
[' C', ' B', ' D'], support: 0.1
['A', ' B', ' D'], support: 0.1
[' C', 'A', ' B', ' D'], support: 0.1
[' C', 'B'], support: 0.1
['B', ' F'], support: 0.15
['B', ' D'], support: 0.1
['B', ' F', ' D'], support: 0.1
[' I', ' F'], support: 0.1
[' F', ' D'], support: 0.1
[' H', ' F'], support: 0.1
['E', ' F'], support: 0.1
```

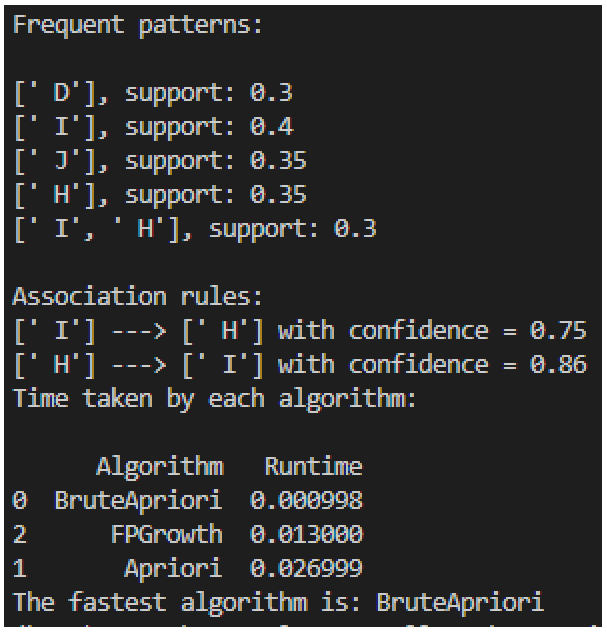


```
Association rules:
['A'] ---> [' D'] with confidence = 0.40
[' D'] ---> ['A'] with confidence = 0.33
['A'] ---> [' J'] with confidence = 0.40
[' J'] ---> ['A'] with confidence = 0.29
[' C'] ---> ['A'] with confidence = 0.60
['A'] ---> [' C'] with confidence = 0.60
[' C'] ---> [' D'] with confidence = 0.60
[' D'] ---> [' C'] with confidence = 0.50
[' C', 'A'] ---> [' D'] with confidence = 0.67
[' C', ' D'] ---> ['A'] with confidence = 0.67
['A', ' D'] ---> [' C'] with confidence = 1.00
[' C'] ---> ['A', ' D'] with confidence = 0.40
['A'] ---> [' C', ' D'] with confidence = 0.40
[' D'] ---> [' C', 'A'] with confidence = 0.33
[' C'] ---> [' B'] with confidence = 0.60
[' B'] ---> [' C'] with confidence = 1.00
['A'] ---> [' B'] with confidence = 0.60
[' B'] ---> ['A'] with confidence = 1.00
[' B'] ---> [' D'] with confidence = 0.67
[' D'] ---> [' B'] with confidence = 0.33
[' C', 'A'] ---> [' B'] with confidence = 1.00
[' C', ' B'] ---> ['A'] with confidence = 1.00
[' B', 'A'] ---> [' C'] with confidence = 1.00
[' C'] ---> [' B', 'A'] with confidence = 0.60
['A'] ---> [' C', ' B'] with confidence = 0.60
[' B'] ---> [' C', 'A'] with confidence = 1.00
[' C', ' B'] ---> [' D'] with confidence = 0.67
[' C', ' D'] ---> [' B'] with confidence = 0.67
[' B', ' D'] ---> [' C'] with confidence = 1.00
```



```
      Algorithm   Runtime
2      FPGrowth  0.010998
1       Apriori  0.020999
0  BruteApriori  0.023011
The fastest algorithm is: FPGrowth
```

13. Support: 0.3 (30%), Confidence: 0.5 (50%)



```
Frequent patterns:

[' D'], support: 0.3
[' I'], support: 0.4
[' J'], support: 0.35
[' H'], support: 0.35
[' I', ' H'], support: 0.3

Association rules:
[' I'] ---> [' H'] with confidence = 0.75
[' H'] ---> [' I'] with confidence = 0.86
Time taken by each algorithm:

      Algorithm   Runtime
0  BruteApriori  0.000998
2      FPGrowth  0.013000
1       Apriori  0.026999
The fastest algorithm is: BruteApriori
```

9. Performance Analysis

The program implements a brute-force method alongside the **mlxtend** package's Apriori and FP-Growth algorithms to validate the results and compare performance times:

- **Brute-Forced Apriori**: Custom implementation to understand the underlying process of the Apriori algorithm.
- **Apriori (mlxtend)**: Utilizes the **mlxtend** implementation for efficient computation of frequent itemsets and association rules.
- **FP-Growth (mlxtend)**: Employs the FP-Growth algorithm for potentially faster performance in finding frequent itemsets, especially beneficial for large datasets.

Results:

The output includes the frequent itemsets and the derived association rules that meet the specified minimum support and confidence levels. Each rule is accompanied by its confidence level, providing insight into the strength of the association.

Conclusion:

The program effectively demonstrates the application of association rule mining in analyzing transaction data to uncover patterns and relationships between items. The use of both Apriori and FP-Growth algorithms showcases the versatility in approach depending on the dataset size and complexity. Performance comparison highlights the efficiency of the **mlxtend** implementations, making them suitable for practical applications in data mining projects.

Recommendations:

For optimal performance and results, users are advised to adjust the minimum support and confidence thresholds based on the dataset characteristics and the specific goals of the analysis. Larger datasets may benefit from the FP-Growth algorithm due to its efficiency in handling dense data structures.

Steps to run the Program:

Minimum requirements to run the program:

- **CPU**: Intel Core i3 or equivalent AMD
- **RAM**: 4GB (8GB recommended for smoother performance)
- **Storage**: 256GB SSD (for faster read/write speeds)
- **OS**: Windows 10, macOS, or a modern Linux distribution
- **Software**: Latest version of Python, Jupyter Notebook or Visual Studio Code

STEP 1: Clone the repository.

Clone the repository from the repository to your local machine using Git. You can do this by running the following command in your terminal:

```
git clone
https://github.com/Bhumireddy2001/Bhumireddy_Sai-Teja-Reddy_Midterm-
```

STEP 2: Create Virtual Environment.

Create a conda environment after opening the repository. It is recommended to do this project in a virtual environment to avoid conflicts with other Python packages you may have installed.

```
conda create -n myenv python -y
conda activate myenv
```

or

```
python -m venv env
source env/bin/activate
```

Note: On Windows, use `env\Scripts\activate`

STEP 3: Install required libraries.

install the requirements!

```
pip install -r requirements.txt
```

STEP 4: Run the python file.

You can use either `python apriori.py`

or

run it using jupyter notebook by opening the `Apriori.ipynb` file.

Note: Please make sure you have installed the required packages before running the code. If any package is missing, please install it from the requirements.txt file.

python apriori.py

Follow the prompts: The script will prompt you to enter a number corresponding to the dataset you want to load. Enter a number between 1 and 5.

Enter the minimum confidence: The script will then prompt you to enter the minimum confidence for the association rules. Enter a valid floating-point number (e.g., 0.1).

Conclusion

This report summarizes the application and comparison of Brute forced Apriori, Apriori and FP-Growth algorithms for transaction data analysis. FP-Growth's efficiency, especially in terms of runtime, demonstrates its suitability for large-scale data analysis. The insights gained from this analysis have significant implications for understanding consumer behavior and optimizing retail strategies.

References and Links:

Note to the Grader: I have created a repository using my personal account. Because my college email is linked with it. If I remove that mail id, I will lose all the benefits of GitHub pro and student benefits. So, please consider this. Thank You.

GitHub Repository link:

https://github.com/Bhumireddy2001/Bhumireddy_Sai-Teja-Reddy_Midterm-Project.git

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation.