NAME: VIJAYA DURGA BHAVANI THOTA

NJIT UCID: vt299

Email Address: vt299@njit.edu

Professor: Yasser Abdullah

COURSE CS634101 Data Mining

Midterm Project Report

Abstract

In this project, I implemented and compared three algorithms for association rule mining: Brute Force, Apriori, and FP-Growth. Each algorithm was tested on transactional datasets to identify frequent itemsets and association rules, with the results assessed for performance and accuracy.

Introduction

Data mining plays a crucial role in extracting meaningful patterns and insights from large datasets. A key technique in this field is association rule mining, which reveals relationships in transactional data and is commonly used in market basket analysis and recommendation systems. In this project, I implemented three algorithms: Brute Force, Apriori, and FP-Growth, to identify frequent itemsets and create association rules. The Brute Force method examines all possible item combinations but is resource-intensive. In contrast, the Apriori Algorithm improves efficiency by leveraging the principle that all subsets of frequent itemsets must also be frequent. The FP-Growth Algorithm further enhances this process by utilizing an FP-Tree, allowing for the mining of frequent itemsets without generating candidate sets. By applying these algorithms to transactional datasets from retailers like Amazon and Costco, I evaluated their performance in terms of execution time, accuracy, and scalability.

In this implementation, I applied the Brute Force, Apriori, and FP-Growth algorithms to custom transactional datasets from various retail stores, including Amazon, Best Buy, Costco, Nike, and

Walgreens. The goal was to uncover frequent itemsets and generate association rules. Key steps in this process included:

Core Concepts and Principles

Frequent Itemset Discovery:

All three algorithms are designed to identify frequent itemsets, which help uncover customer purchasing patterns

Support and Confidence:

Support and confidence are two important metrics in data mining. Support indicates the frequency with which an item or itemset appears, while confidence evaluates the probability of items being bought together. These metrics play a crucial role in directing our analysis.

Association Rules:

By identifying strong association rules, I can pinpoint which items are frequently bought together. These rules are essential for enhancing sales strategies, including product recommendations.

Project Setup:

1. Clone the github repository from link : **https://github.com/vijji918/DataMining_MidTerm_Project/tree/main**
2. Go to project directory
3. Install python libraries by executing **pip install -r requirements.txt**
4. **Run command "python .\mid_term_project.py" to start the program**

Tools and Libraries:

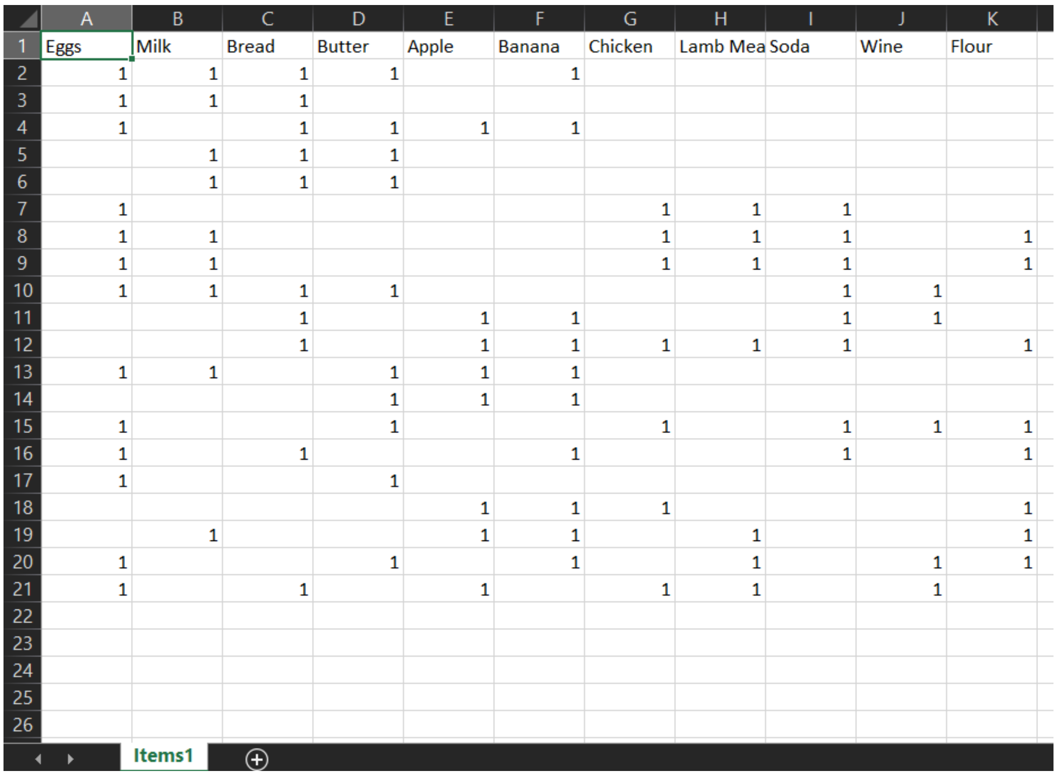
To execute the program we need following libraries:

1. mlxtend
2. pandas
3. numpy
4. itertools

Data:

For testing the implementation I have created 5 datasets with 20 transactions and 10 unique items.

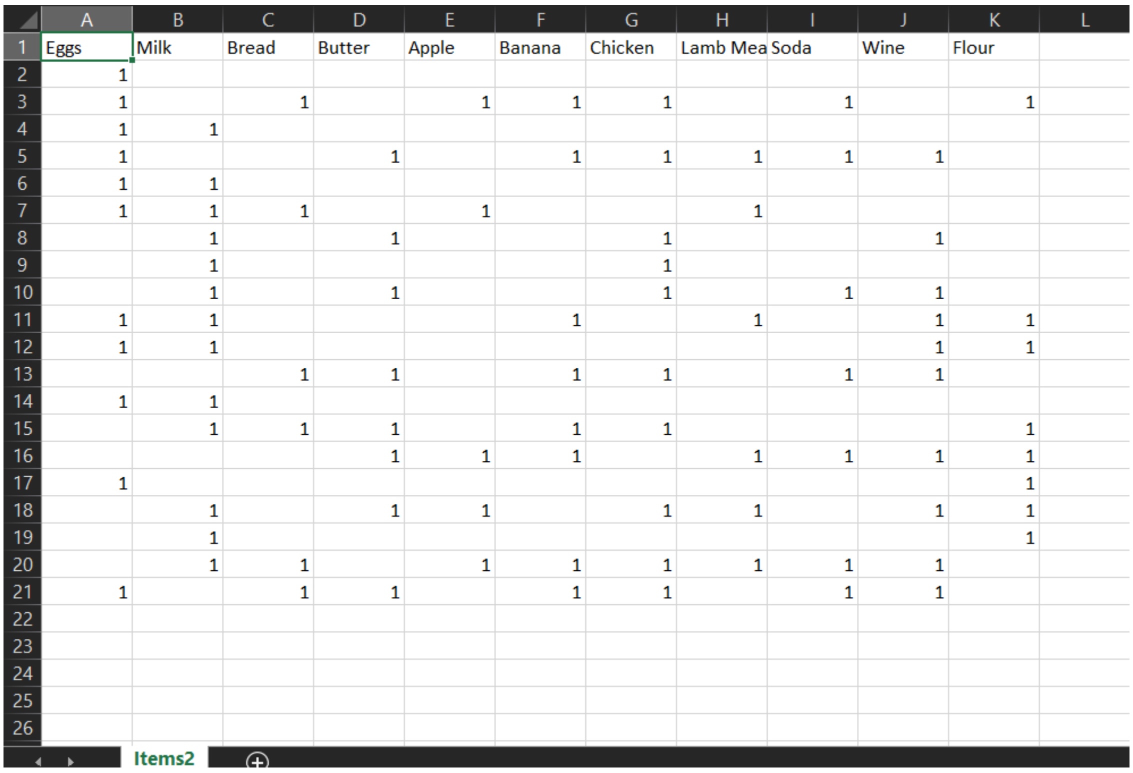
Items1.csv



	A	B	C	D	E	F	G	H	I	J	K
1	Eggs	Milk	Bread	Butter	Apple	Banana	Chicken	Lamb Mea	Soda	Wine	Flour
2	1	1	1	1		1					
3	1	1	1								
4	1		1	1	1	1					
5		1	1	1							
6		1	1	1							
7	1						1	1	1		
8	1	1					1	1	1		1
9	1	1					1	1	1		1
10	1	1	1	1					1	1	
11			1		1	1			1	1	
12			1		1	1	1	1	1		1
13	1	1		1	1	1					
14				1	1	1					
15	1			1			1		1	1	1
16	1		1			1			1		1
17	1			1							
18					1	1	1				1
19		1			1	1		1			1
20	1			1		1		1		1	1
21	1		1		1		1	1		1	
22											
23											
24											
25											
26											

Items1

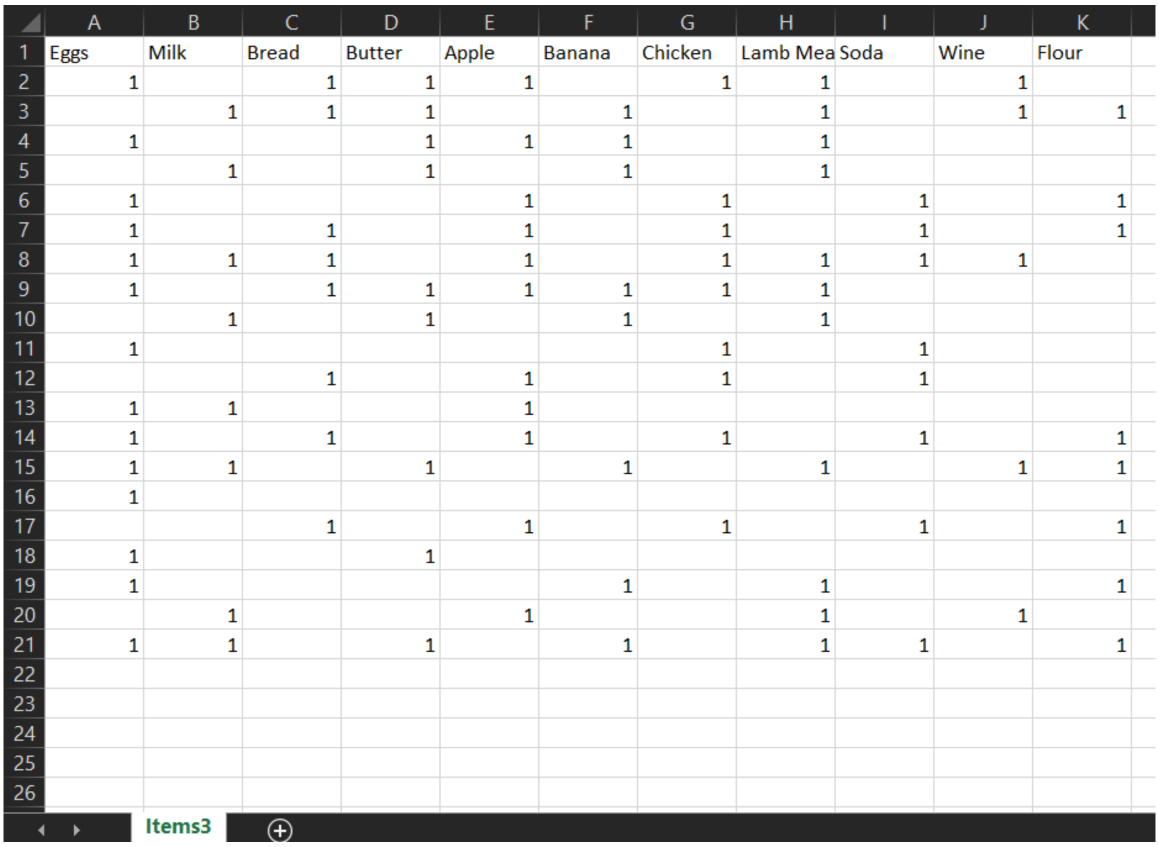
Items2.csv



	A	B	C	D	E	F	G	H	I	J	K	L
1	Eggs	Milk	Bread	Butter	Apple	Banana	Chicken	Lamb Mea	Soda	Wine	Flour	
2	1											
3	1		1		1	1	1		1		1	
4	1	1										
5	1			1		1	1	1	1	1		
6	1	1										
7	1	1	1		1			1				
8		1		1			1			1		
9		1					1					
10		1		1			1		1	1		
11	1	1				1		1		1	1	
12	1	1								1	1	
13			1	1		1	1		1	1		
14	1	1										
15		1	1	1		1	1				1	
16				1	1	1		1	1	1	1	
17	1										1	
18		1		1	1		1	1		1	1	
19		1									1	
20		1	1		1	1	1	1	1	1		
21	1		1	1		1	1		1	1		
22												
23												
24												
25												
26												

Items2

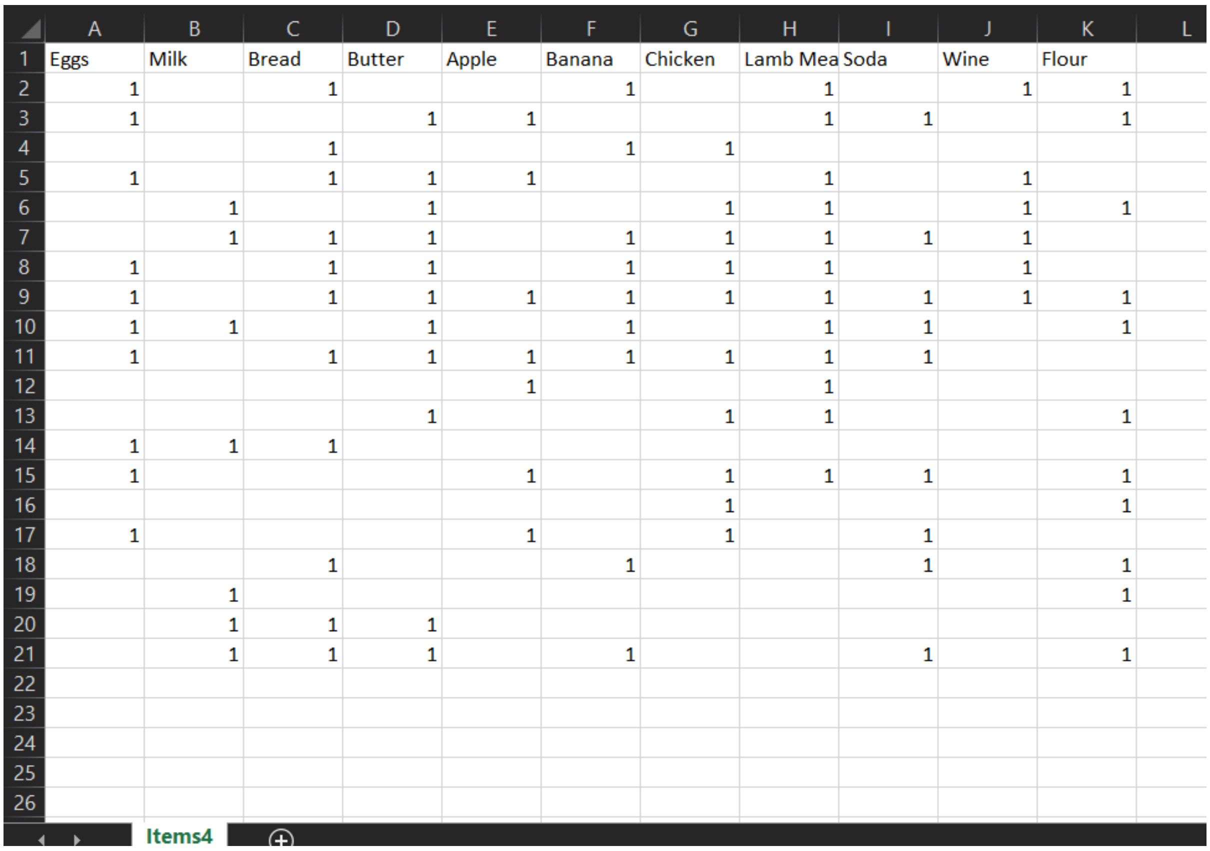
Items3.csv



	A	B	C	D	E	F	G	H	I	J	K
1	Eggs	Milk	Bread	Butter	Apple	Banana	Chicken	Lamb Mea	Soda	Wine	Flour
2	1		1	1	1		1	1		1	
3		1	1	1		1		1		1	1
4	1			1	1	1		1			
5		1		1		1		1			
6	1				1		1		1		1
7	1		1		1		1		1		1
8	1	1	1		1		1	1	1	1	
9	1		1	1	1	1	1	1			
10		1		1		1		1			
11	1						1		1		
12			1		1		1		1		
13	1	1			1						
14	1		1		1		1		1		1
15	1	1		1		1		1		1	1
16	1										
17			1		1		1		1		1
18	1			1							
19	1					1		1			1
20		1			1			1		1	
21	1	1		1		1		1	1		1
22											
23											
24											
25											
26											

Items3

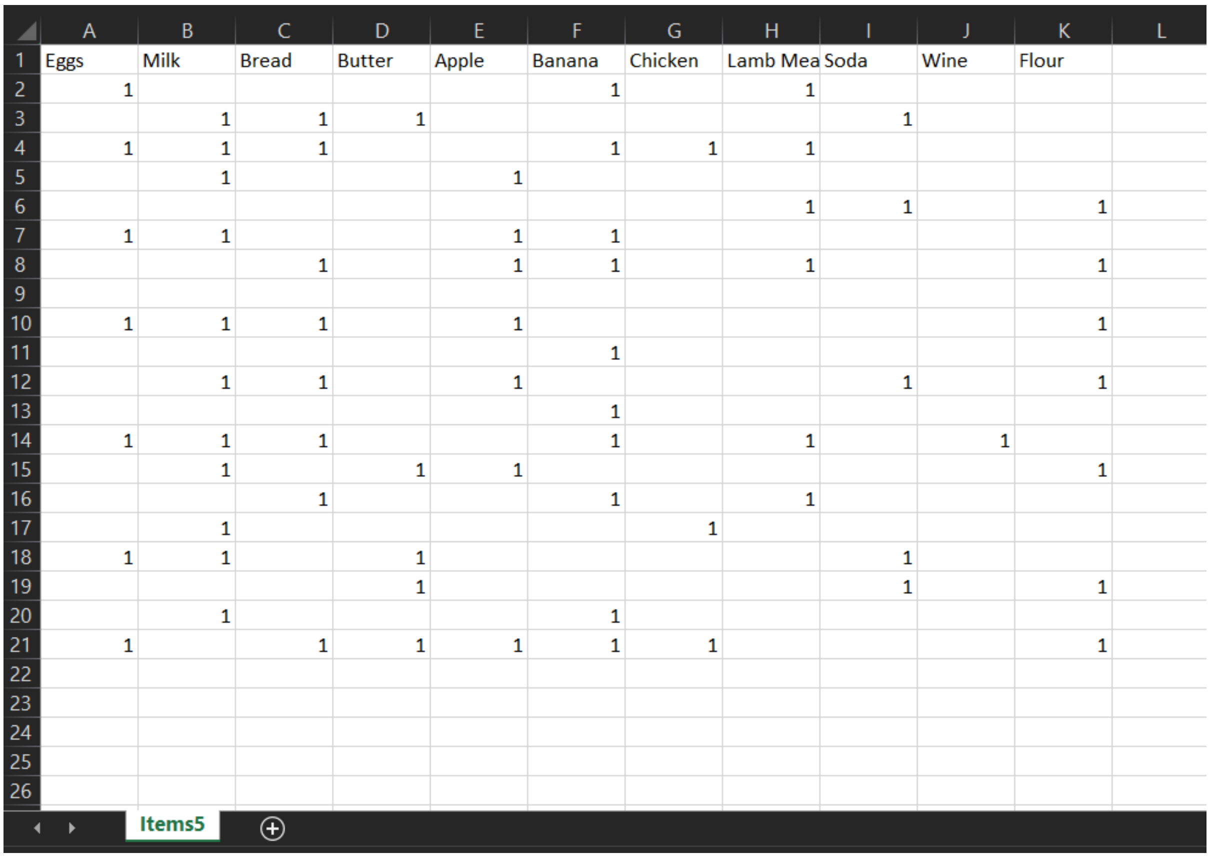
Items4.csv



	A	B	C	D	E	F	G	H	I	J	K	L
1	Eggs	Milk	Bread	Butter	Apple	Banana	Chicken	Lamb Mea	Soda	Wine	Flour	
2	1		1			1		1		1	1	
3	1			1	1			1	1		1	
4			1			1	1					
5	1		1	1	1			1		1		
6		1		1			1	1		1	1	
7		1	1	1		1	1	1	1	1		
8	1		1	1		1	1	1		1		
9	1		1	1	1	1	1	1	1	1	1	
10	1	1		1		1		1	1		1	
11	1		1	1	1	1	1	1	1			
12					1			1				
13				1			1	1			1	
14	1	1	1									
15	1				1		1	1	1		1	
16							1				1	
17	1				1		1		1			
18			1			1			1		1	
19		1									1	
20		1	1	1								
21		1	1	1		1			1		1	
22												
23												
24												
25												
26												

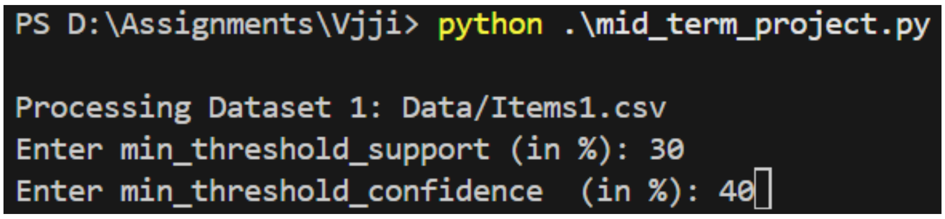
Items4

Items5.csv

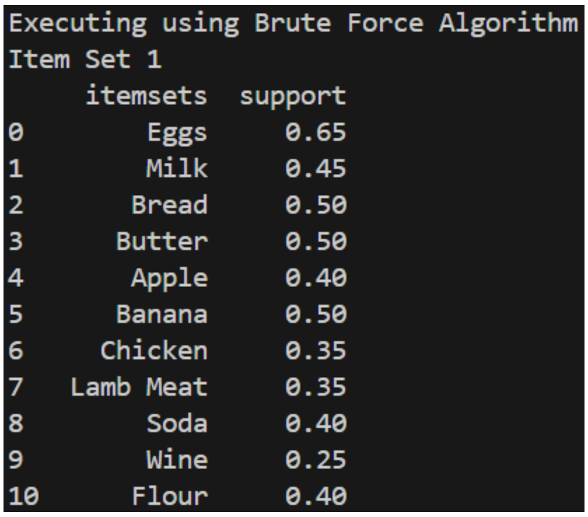


	A	B	C	D	E	F	G	H	I	J	K	L
1	Eggs	Milk	Bread	Butter	Apple	Banana	Chicken	Lamb Mea	Soda	Wine	Flour	
2	1					1		1				
3		1	1	1					1			
4	1	1	1			1	1	1				
5		1			1							
6								1	1		1	
7	1	1			1	1						
8			1		1	1		1			1	
9												
10	1	1	1		1						1	
11						1						
12		1	1		1				1		1	
13						1						
14	1	1	1			1		1		1		
15		1		1	1						1	
16			1			1		1				
17		1					1					
18	1	1		1					1			
19				1					1		1	
20		1				1						
21	1		1	1	1	1	1				1	
22												
23												
24												
25												
26												

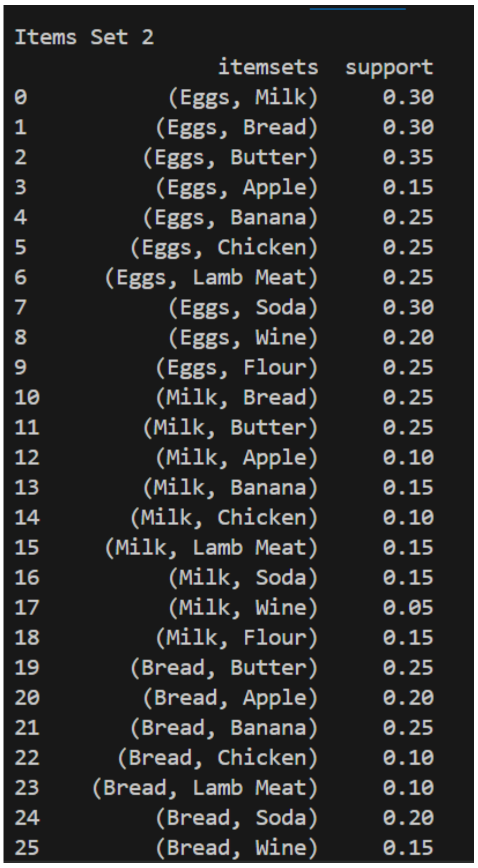
Working:



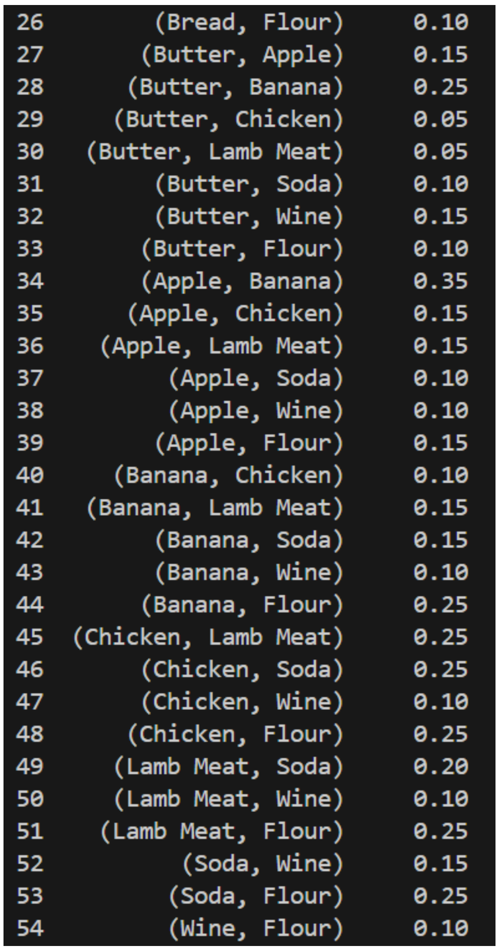
```
PS D:\Assignments\Vjji> python .\mid_term_project.py

Processing Dataset 1: Data/Items1.csv
Enter min_threshold_support (in %): 30
Enter min_threshold_confidence  (in %): 40
```

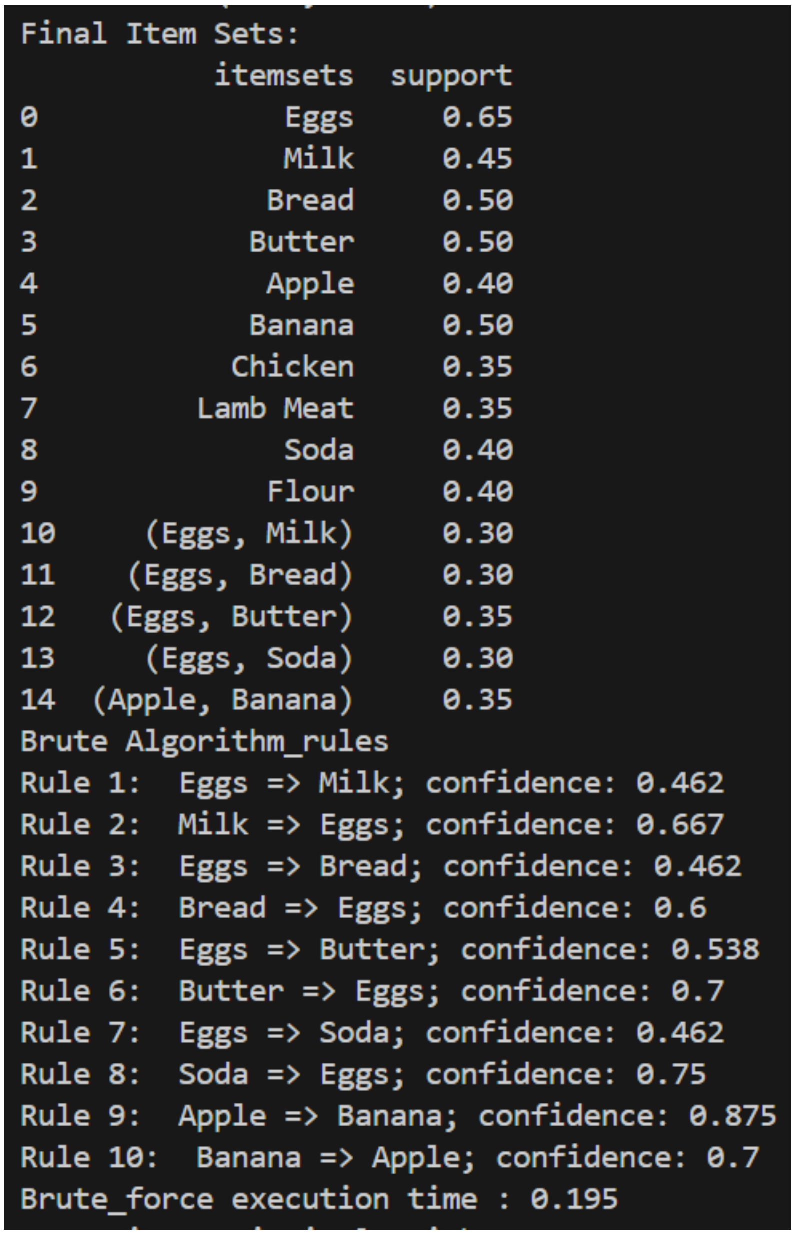


```
Executing using Brute Force Algorithm
Item Set 1
     itemsets  support
0        Eggs     0.65
1        Milk     0.45
2       Bread     0.50
3      Butter     0.50
4       Apple     0.40
5      Banana     0.50
6     Chicken     0.35
7   Lamb Meat     0.35
8        Soda     0.40
9        Wine     0.25
10      Flour     0.40
```

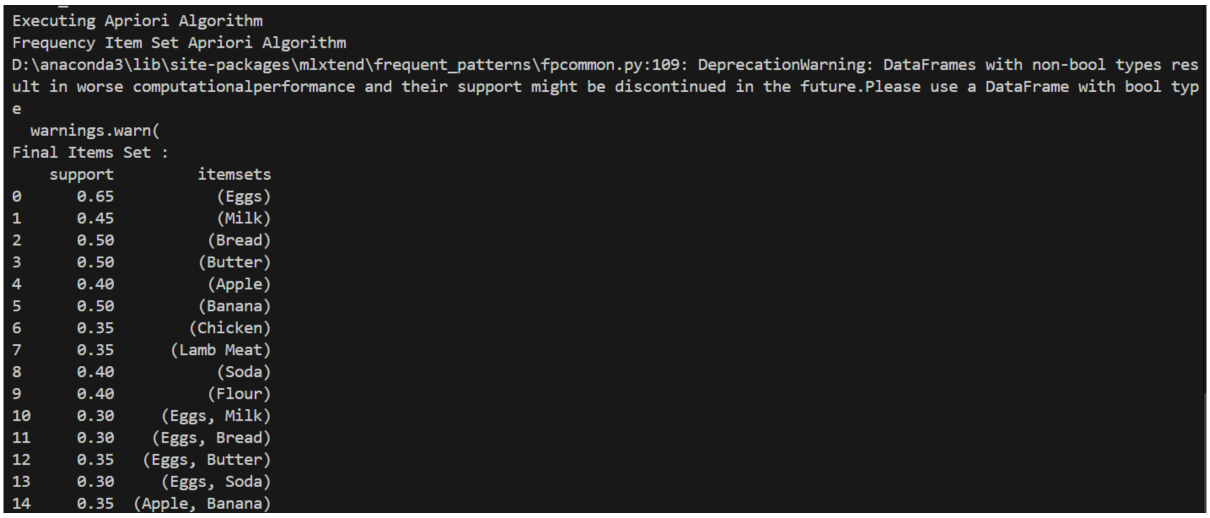


```
Items Set 2
                itemsets  support
0           (Eggs, Milk)     0.30
1          (Eggs, Bread)     0.30
2         (Eggs, Butter)     0.35
3          (Eggs, Apple)     0.15
4         (Eggs, Banana)     0.25
5        (Eggs, Chicken)     0.25
6      (Eggs, Lamb Meat)     0.25
7           (Eggs, Soda)     0.30
8           (Eggs, Wine)     0.20
9          (Eggs, Flour)     0.25
10         (Milk, Bread)     0.25
11        (Milk, Butter)     0.25
12         (Milk, Apple)     0.10
13        (Milk, Banana)     0.15
14       (Milk, Chicken)     0.10
15     (Milk, Lamb Meat)     0.15
16          (Milk, Soda)     0.15
17          (Milk, Wine)     0.05
18         (Milk, Flour)     0.15
19       (Bread, Butter)     0.25
20        (Bread, Apple)     0.20
21       (Bread, Banana)     0.25
22      (Bread, Chicken)     0.10
23    (Bread, Lamb Meat)     0.10
24         (Bread, Soda)     0.20
25         (Bread, Wine)     0.15
```

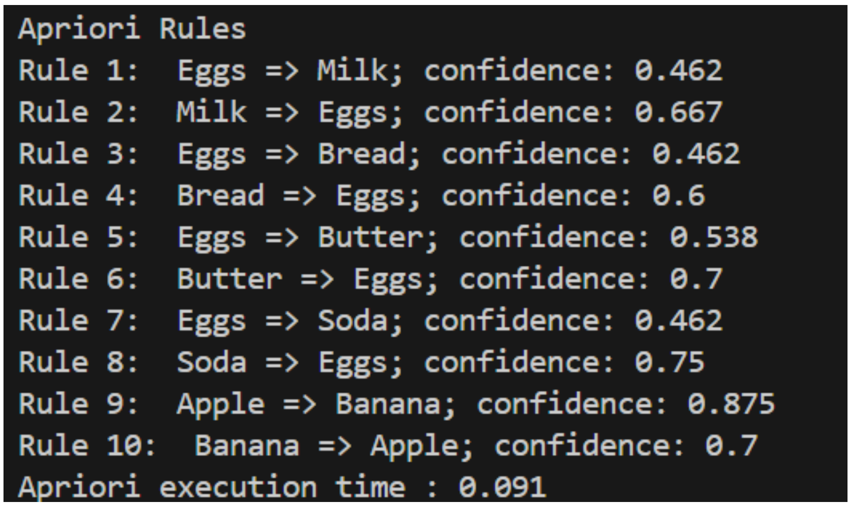

```
26         (Bread, Flour)     0.10
27        (Butter, Apple)     0.15
28       (Butter, Banana)     0.25
29      (Butter, Chicken)     0.05
30    (Butter, Lamb Meat)     0.05
31         (Butter, Soda)     0.10
32         (Butter, Wine)     0.15
33        (Butter, Flour)     0.10
34        (Apple, Banana)     0.35
35       (Apple, Chicken)     0.15
36     (Apple, Lamb Meat)     0.15
37          (Apple, Soda)     0.10
38          (Apple, Wine)     0.10
39         (Apple, Flour)     0.15
40      (Banana, Chicken)     0.10
41    (Banana, Lamb Meat)     0.15
42         (Banana, Soda)     0.15
43         (Banana, Wine)     0.10
44        (Banana, Flour)     0.25
45   (Chicken, Lamb Meat)     0.25
46        (Chicken, Soda)     0.25
47        (Chicken, Wine)     0.10
48       (Chicken, Flour)     0.25
49      (Lamb Meat, Soda)     0.20
50      (Lamb Meat, Wine)     0.10
51     (Lamb Meat, Flour)     0.25
52           (Soda, Wine)     0.15
53          (Soda, Flour)     0.25
54          (Wine, Flour)     0.10
```



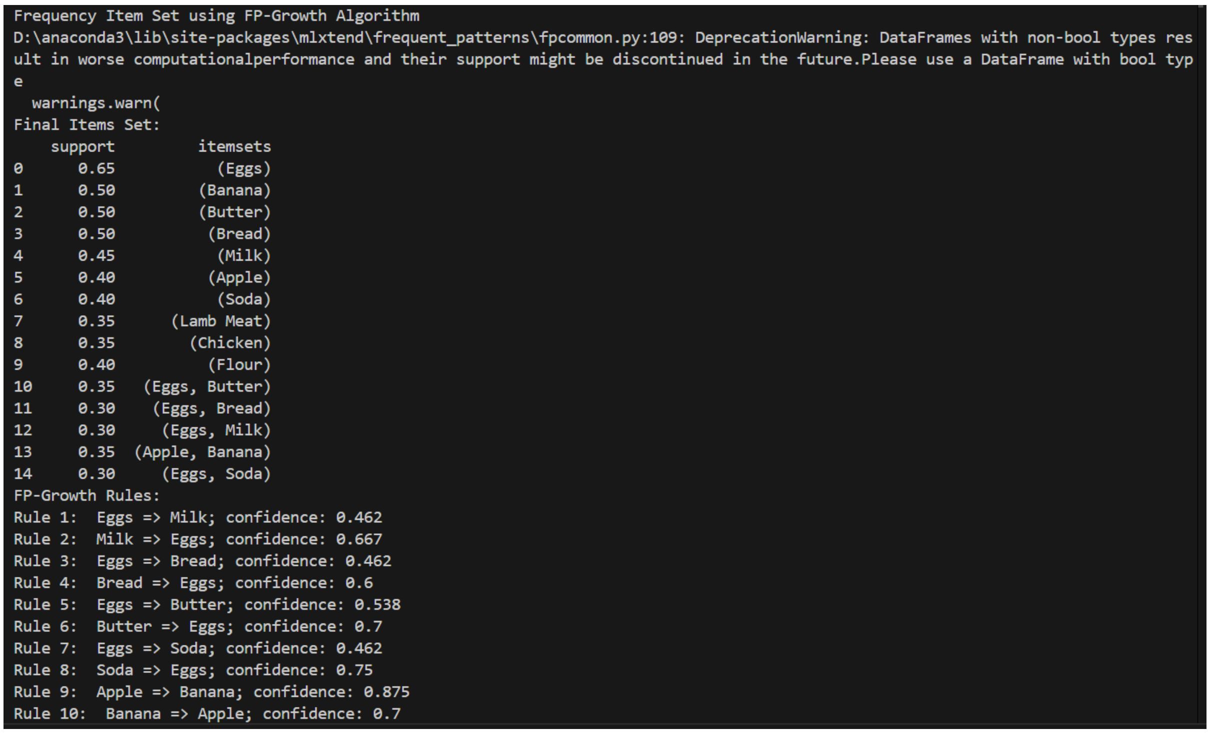
```
Final Item Sets:
           itemsets  support
0              Eggs     0.65
1              Milk     0.45
2             Bread     0.50
3            Butter     0.50
4             Apple     0.40
5            Banana     0.50
6           Chicken     0.35
7         Lamb Meat     0.35
8              Soda     0.40
9             Flour     0.40
10     (Eggs, Milk)     0.30
11    (Eggs, Bread)     0.30
12   (Eggs, Butter)     0.35
13     (Eggs, Soda)     0.30
14  (Apple, Banana)     0.35
Brute Algorithm_rules
Rule 1:  Eggs => Milk; confidence: 0.462
Rule 2:  Milk => Eggs; confidence: 0.667
Rule 3:  Eggs => Bread; confidence: 0.462
Rule 4:  Bread => Eggs; confidence: 0.6
Rule 5:  Eggs => Butter; confidence: 0.538
Rule 6:  Butter => Eggs; confidence: 0.7
Rule 7:  Eggs => Soda; confidence: 0.462
Rule 8:  Soda => Eggs; confidence: 0.75
Rule 9:  Apple => Banana; confidence: 0.875
Rule 10:  Banana => Apple; confidence: 0.7
Brute_force execution time : 0.195
```



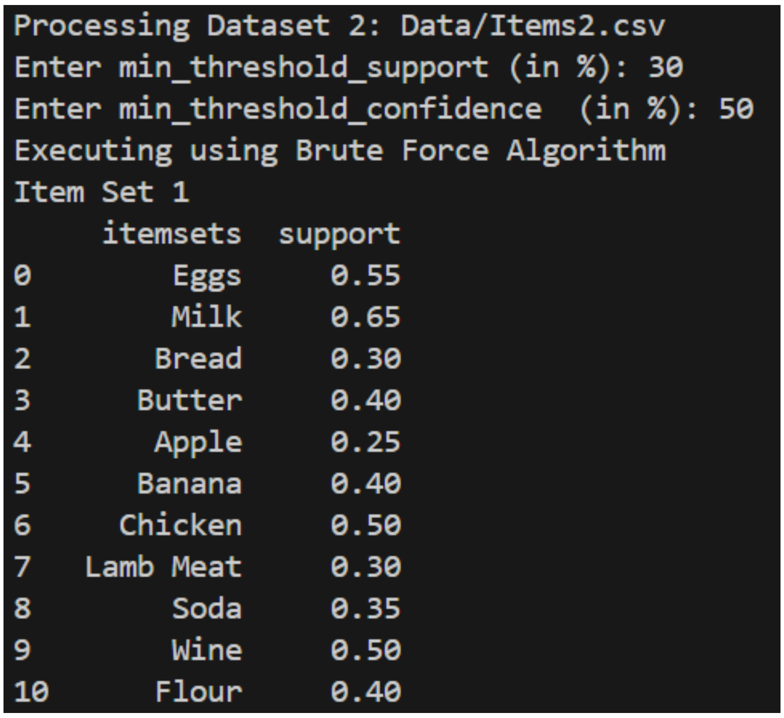
```
Executing Apriori Algorithm
Frequency Item Set Apriori Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with non-bool types res
ult in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool typ
e
  warnings.warn(
Final Items Set :
    support         itemsets
0      0.65           (Eggs)
1      0.45           (Milk)
2      0.50          (Bread)
3      0.50         (Butter)
4      0.40          (Apple)
5      0.50         (Banana)
6      0.35        (Chicken)
7      0.35      (Lamb Meat)
8      0.40           (Soda)
9      0.40          (Flour)
10     0.30     (Eggs, Milk)
11     0.30    (Eggs, Bread)
12     0.35   (Eggs, Butter)
13     0.30     (Eggs, Soda)
14     0.35  (Apple, Banana)
```



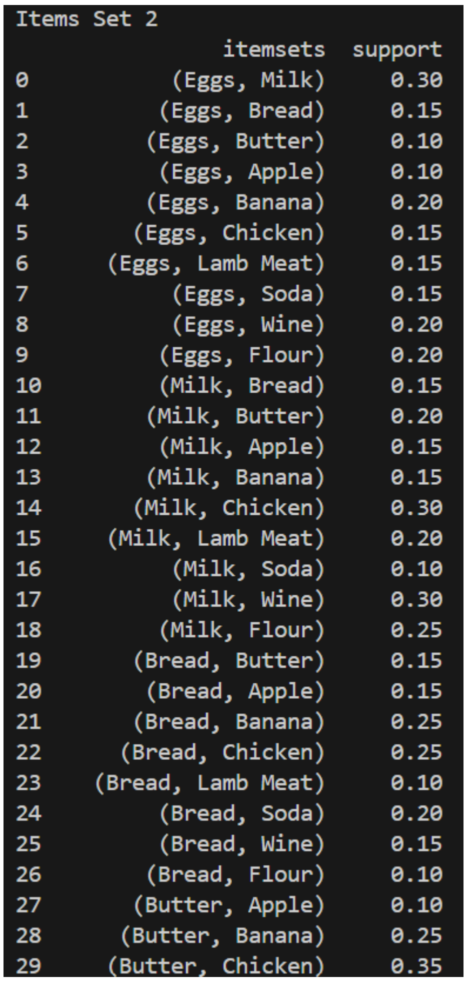
```
Apriori Rules
Rule 1:  Eggs => Milk; confidence: 0.462
Rule 2:  Milk => Eggs; confidence: 0.667
Rule 3:  Eggs => Bread; confidence: 0.462
Rule 4:  Bread => Eggs; confidence: 0.6
Rule 5:  Eggs => Butter; confidence: 0.538
Rule 6:  Butter => Eggs; confidence: 0.7
Rule 7:  Eggs => Soda; confidence: 0.462
Rule 8:  Soda => Eggs; confidence: 0.75
Rule 9:  Apple => Banana; confidence: 0.875
Rule 10:  Banana => Apple; confidence: 0.7
Apriori execution time : 0.091
```



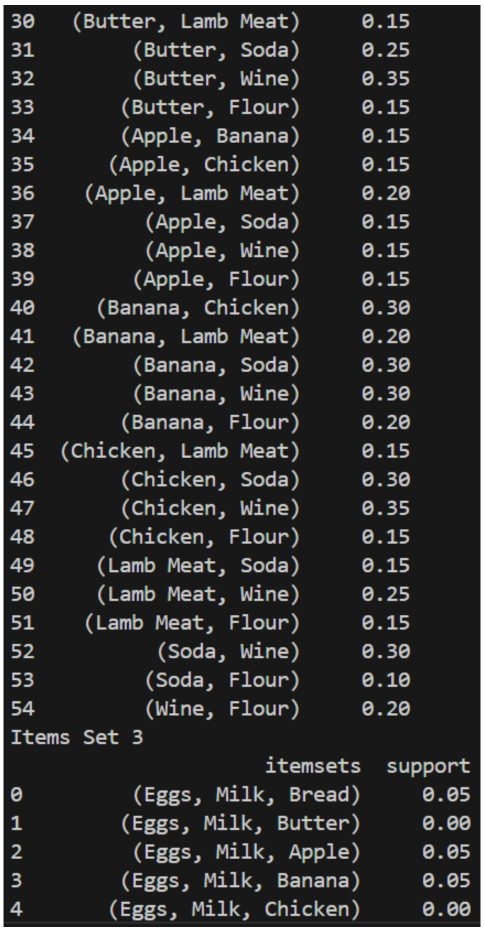
```
Frequency Item Set using FP-Growth Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with non-bool types res
ult in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool typ
e
  warnings.warn(
Final Items Set:
    support         itemsets
0      0.65           (Eggs)
1      0.50         (Banana)
2      0.50         (Butter)
3      0.50          (Bread)
4      0.45           (Milk)
5      0.40          (Apple)
6      0.40           (Soda)
7      0.35      (Lamb Meat)
8      0.35        (Chicken)
9      0.40          (Flour)
10     0.35   (Eggs, Butter)
11     0.30    (Eggs, Bread)
12     0.30     (Eggs, Milk)
13     0.35  (Apple, Banana)
14     0.30     (Eggs, Soda)
FP-Growth Rules:
Rule 1:  Eggs => Milk; confidence: 0.462
Rule 2:  Milk => Eggs; confidence: 0.667
Rule 3:  Eggs => Bread; confidence: 0.462
Rule 4:  Bread => Eggs; confidence: 0.6
Rule 5:  Eggs => Butter; confidence: 0.538
Rule 6:  Butter => Eggs; confidence: 0.7
Rule 7:  Eggs => Soda; confidence: 0.462
Rule 8:  Soda => Eggs; confidence: 0.75
Rule 9:  Apple => Banana; confidence: 0.875
Rule 10:  Banana => Apple; confidence: 0.7
```



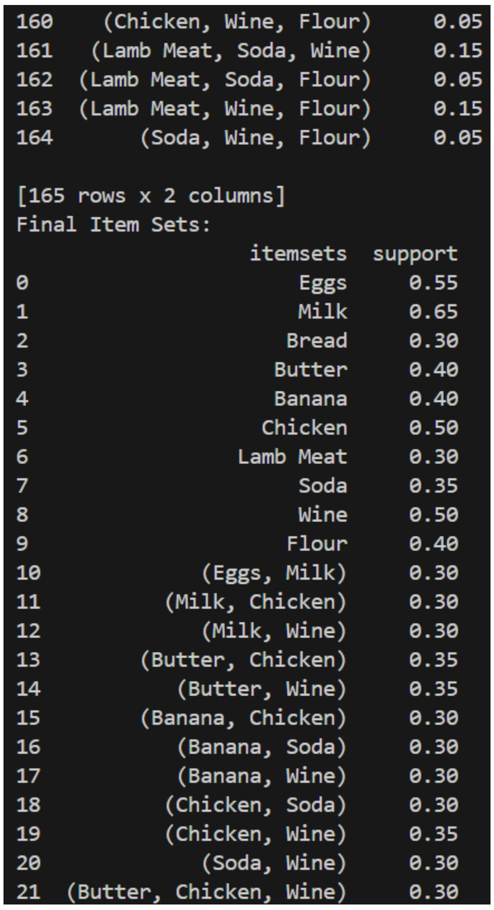
```
Processing Dataset 2: Data/Items2.csv
Enter min_threshold_support (in %): 30
Enter min_threshold_confidence  (in %): 50
Executing using Brute Force Algorithm
Item Set 1
     itemsets  support
0        Eggs     0.55
1        Milk     0.65
2       Bread     0.30
3      Butter     0.40
4       Apple     0.25
5      Banana     0.40
6     Chicken     0.50
7   Lamb Meat     0.30
8        Soda     0.35
9        Wine     0.50
10      Flour     0.40
```



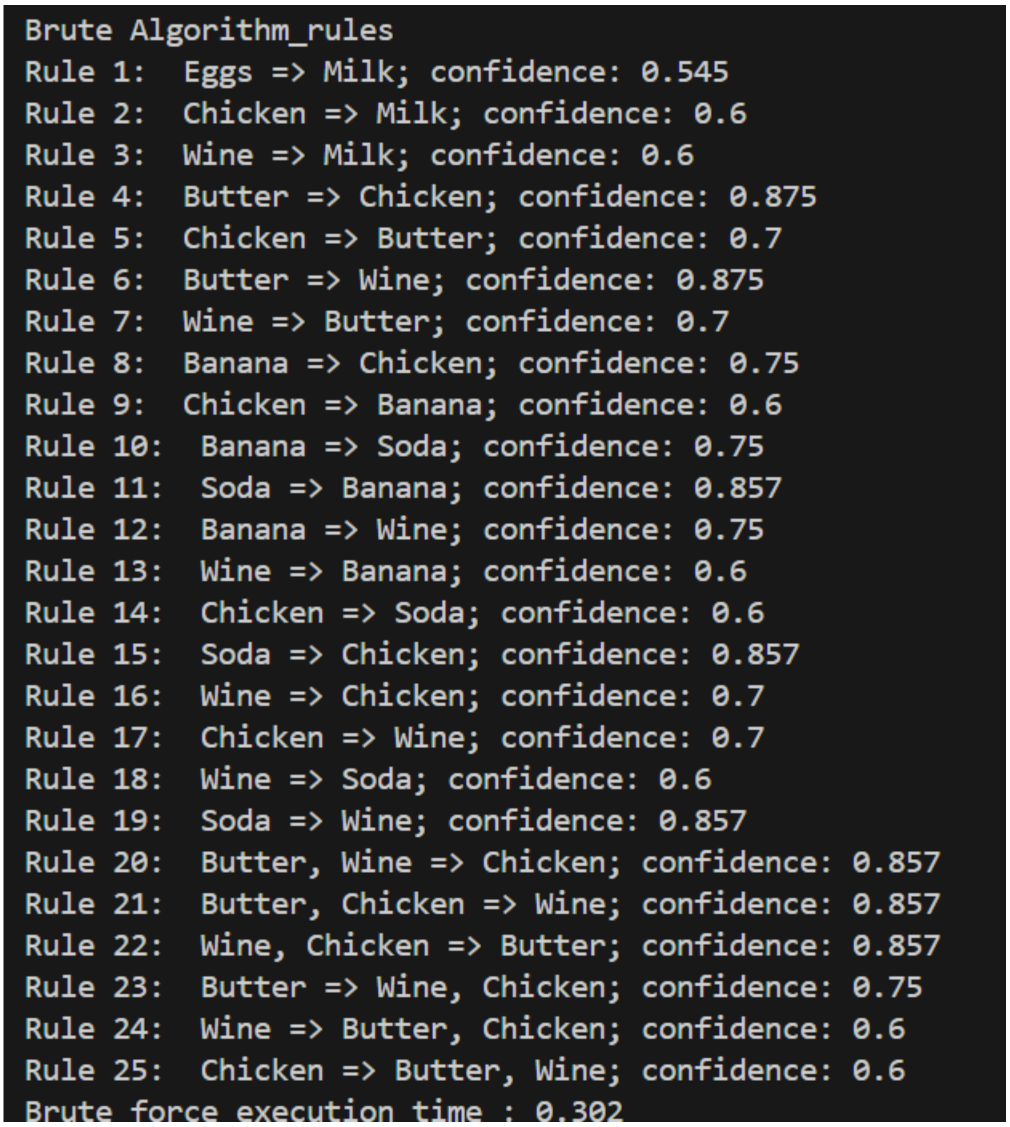
```
Items Set 2
                itemsets  support
0           (Eggs, Milk)     0.30
1          (Eggs, Bread)     0.15
2         (Eggs, Butter)     0.10
3          (Eggs, Apple)     0.10
4         (Eggs, Banana)     0.20
5        (Eggs, Chicken)     0.15
6      (Eggs, Lamb Meat)     0.15
7           (Eggs, Soda)     0.15
8           (Eggs, Wine)     0.20
9          (Eggs, Flour)     0.20
10         (Milk, Bread)     0.15
11        (Milk, Butter)     0.20
12         (Milk, Apple)     0.15
13        (Milk, Banana)     0.15
14       (Milk, Chicken)     0.30
15     (Milk, Lamb Meat)     0.20
16          (Milk, Soda)     0.10
17          (Milk, Wine)     0.30
18         (Milk, Flour)     0.25
19       (Bread, Butter)     0.15
20        (Bread, Apple)     0.15
21       (Bread, Banana)     0.25
22      (Bread, Chicken)     0.25
23    (Bread, Lamb Meat)     0.10
24         (Bread, Soda)     0.20
25         (Bread, Wine)     0.15
26        (Bread, Flour)     0.10
27       (Butter, Apple)     0.10
28      (Butter, Banana)     0.25
29     (Butter, Chicken)     0.35
```



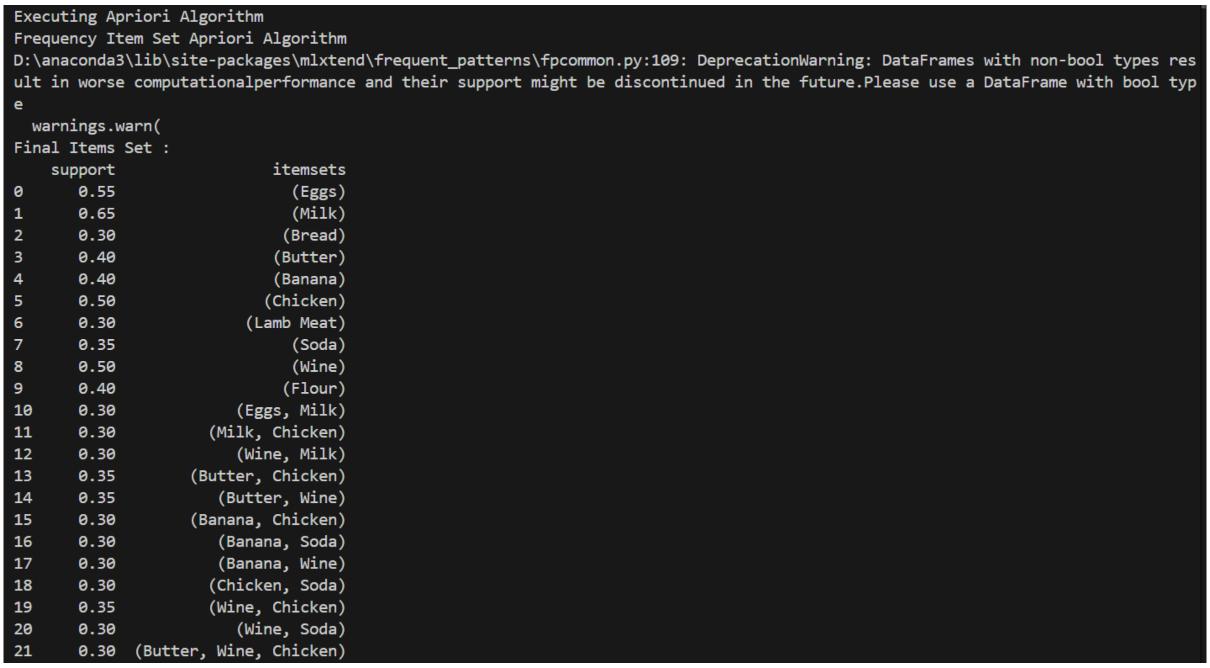
```
30   (Butter, Lamb Meat)      0.15
31        (Butter, Soda)      0.25
32        (Butter, Wine)      0.35
33       (Butter, Flour)      0.15
34       (Apple, Banana)      0.15
35      (Apple, Chicken)      0.15
36    (Apple, Lamb Meat)      0.20
37         (Apple, Soda)      0.15
38         (Apple, Wine)      0.15
39        (Apple, Flour)      0.15
40     (Banana, Chicken)      0.30
41   (Banana, Lamb Meat)      0.20
42        (Banana, Soda)      0.30
43        (Banana, Wine)      0.30
44       (Banana, Flour)      0.20
45  (Chicken, Lamb Meat)      0.15
46       (Chicken, Soda)      0.30
47       (Chicken, Wine)      0.35
48      (Chicken, Flour)      0.15
49     (Lamb Meat, Soda)      0.15
50     (Lamb Meat, Wine)      0.25
51    (Lamb Meat, Flour)      0.15
52          (Soda, Wine)      0.30
53         (Soda, Flour)      0.10
54         (Wine, Flour)      0.20
Items Set 3
                     itemsets  support
0          (Eggs, Milk, Bread)     0.05
1         (Eggs, Milk, Butter)     0.00
2          (Eggs, Milk, Apple)     0.05
3         (Eggs, Milk, Banana)     0.05
4        (Eggs, Milk, Chicken)     0.00
```



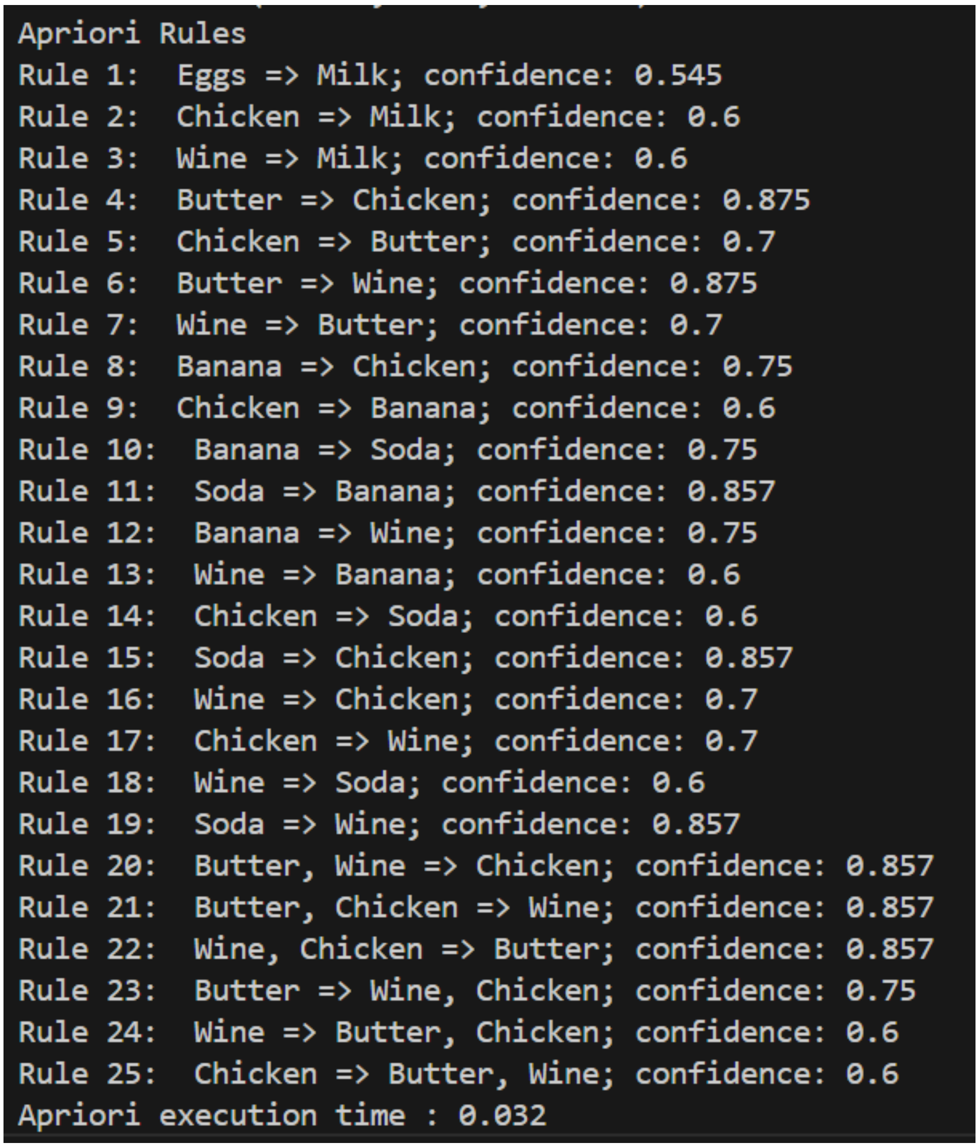
```
160    (Chicken, Wine, Flour)     0.05
161   (Lamb Meat, Soda, Wine)     0.15
162  (Lamb Meat, Soda, Flour)     0.05
163  (Lamb Meat, Wine, Flour)     0.15
164       (Soda, Wine, Flour)     0.05

[165 rows x 2 columns]
Final Item Sets:
                   itemsets  support
0                      Eggs     0.55
1                      Milk     0.65
2                     Bread     0.30
3                    Butter     0.40
4                    Banana     0.40
5                   Chicken     0.50
6                 Lamb Meat     0.30
7                      Soda     0.35
8                      Wine     0.50
9                     Flour     0.40
10             (Eggs, Milk)     0.30
11          (Milk, Chicken)     0.30
12             (Milk, Wine)     0.30
13        (Butter, Chicken)     0.35
14           (Butter, Wine)     0.35
15        (Banana, Chicken)     0.30
16           (Banana, Soda)     0.30
17           (Banana, Wine)     0.30
18          (Chicken, Soda)     0.30
19          (Chicken, Wine)     0.35
20             (Soda, Wine)     0.30
21  (Butter, Chicken, Wine)     0.30
```



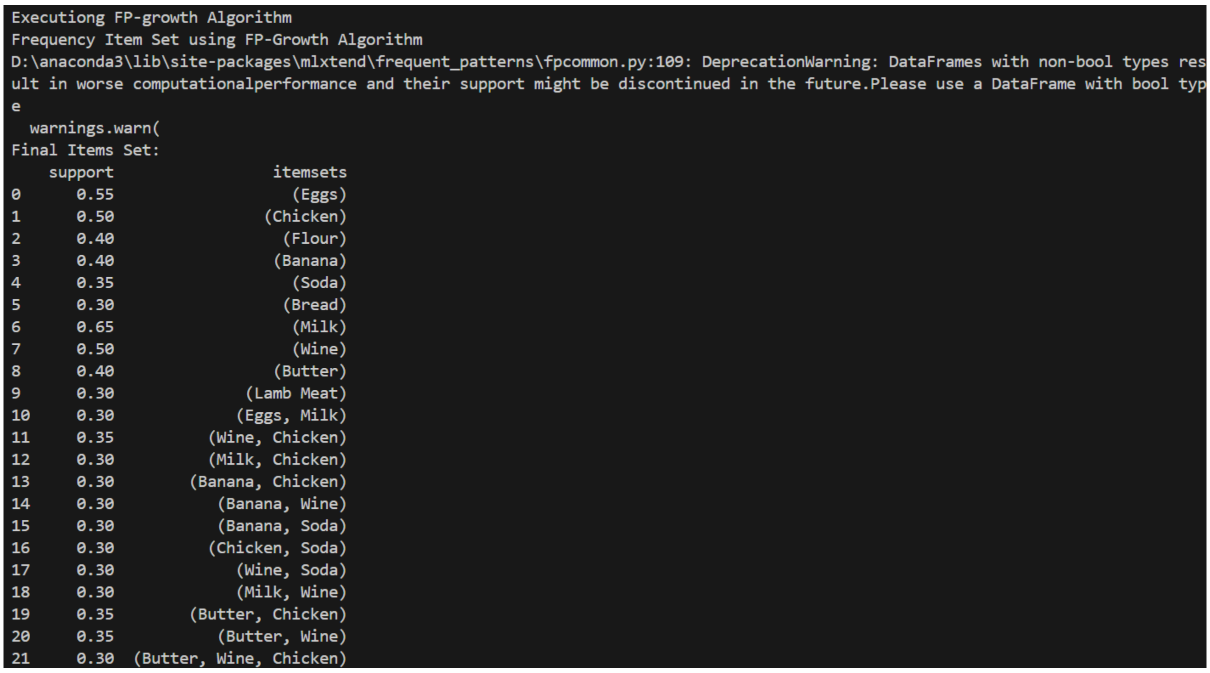
```
Brute Algorithm_rules
Rule 1:  Eggs => Milk; confidence: 0.545
Rule 2:  Chicken => Milk; confidence: 0.6
Rule 3:  Wine => Milk; confidence: 0.6
Rule 4:  Butter => Chicken; confidence: 0.875
Rule 5:  Chicken => Butter; confidence: 0.7
Rule 6:  Butter => Wine; confidence: 0.875
Rule 7:  Wine => Butter; confidence: 0.7
Rule 8:  Banana => Chicken; confidence: 0.75
Rule 9:  Chicken => Banana; confidence: 0.6
Rule 10:  Banana => Soda; confidence: 0.75
Rule 11:  Soda => Banana; confidence: 0.857
Rule 12:  Banana => Wine; confidence: 0.75
Rule 13:  Wine => Banana; confidence: 0.6
Rule 14:  Chicken => Soda; confidence: 0.6
Rule 15:  Soda => Chicken; confidence: 0.857
Rule 16:  Wine => Chicken; confidence: 0.7
Rule 17:  Chicken => Wine; confidence: 0.7
Rule 18:  Wine => Soda; confidence: 0.6
Rule 19:  Soda => Wine; confidence: 0.857
Rule 20:  Butter, Wine => Chicken; confidence: 0.857
Rule 21:  Butter, Chicken => Wine; confidence: 0.857
Rule 22:  Wine, Chicken => Butter; confidence: 0.857
Rule 23:  Butter => Wine, Chicken; confidence: 0.75
Rule 24:  Wine => Butter, Chicken; confidence: 0.6
Rule 25:  Chicken => Butter, Wine; confidence: 0.6
Brute force execution time : 0.302
```



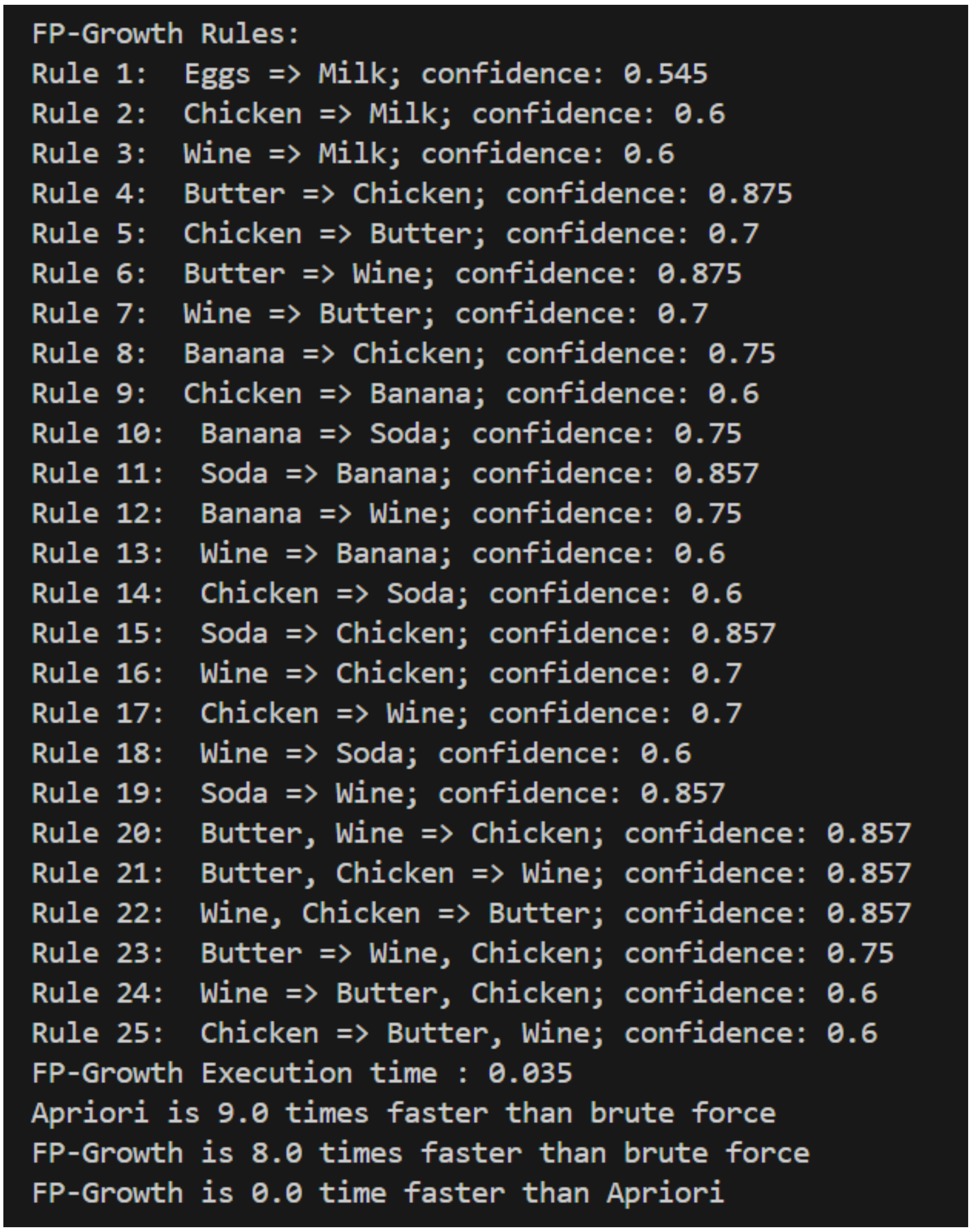
```
Executing Apriori Algorithm
Frequency Item Set Apriori Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with non-bool types res
ult in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool typ
e
  warnings.warn(
Final Items Set :
    support                 itemsets
0      0.55                   (Eggs)
1      0.65                   (Milk)
2      0.30                  (Bread)
3      0.40                 (Butter)
4      0.40                 (Banana)
5      0.50                (Chicken)
6      0.30              (Lamb Meat)
7      0.35                   (Soda)
8      0.50                   (Wine)
9      0.40                  (Flour)
10     0.30             (Eggs, Milk)
11     0.30          (Milk, Chicken)
12     0.30             (Wine, Milk)
13     0.35        (Butter, Chicken)
14     0.35           (Butter, Wine)
15     0.30        (Banana, Chicken)
16     0.30           (Banana, Soda)
17     0.30           (Banana, Wine)
18     0.30          (Chicken, Soda)
19     0.35          (Wine, Chicken)
20     0.30             (Wine, Soda)
21     0.30  (Butter, Wine, Chicken)
```



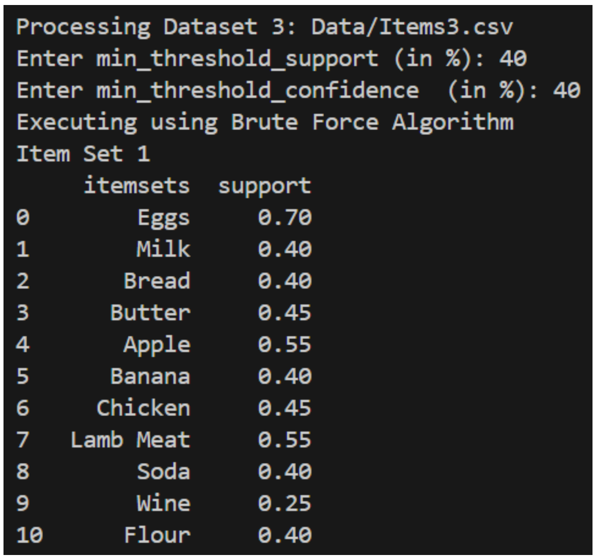
```
Apriori Rules
Rule 1:  Eggs => Milk; confidence: 0.545
Rule 2:  Chicken => Milk; confidence: 0.6
Rule 3:  Wine => Milk; confidence: 0.6
Rule 4:  Butter => Chicken; confidence: 0.875
Rule 5:  Chicken => Butter; confidence: 0.7
Rule 6:  Butter => Wine; confidence: 0.875
Rule 7:  Wine => Butter; confidence: 0.7
Rule 8:  Banana => Chicken; confidence: 0.75
Rule 9:  Chicken => Banana; confidence: 0.6
Rule 10:  Banana => Soda; confidence: 0.75
Rule 11:  Soda => Banana; confidence: 0.857
Rule 12:  Banana => Wine; confidence: 0.75
Rule 13:  Wine => Banana; confidence: 0.6
Rule 14:  Chicken => Soda; confidence: 0.6
Rule 15:  Soda => Chicken; confidence: 0.857
Rule 16:  Wine => Chicken; confidence: 0.7
Rule 17:  Chicken => Wine; confidence: 0.7
Rule 18:  Wine => Soda; confidence: 0.6
Rule 19:  Soda => Wine; confidence: 0.857
Rule 20:  Butter, Wine => Chicken; confidence: 0.857
Rule 21:  Butter, Chicken => Wine; confidence: 0.857
Rule 22:  Wine, Chicken => Butter; confidence: 0.857
Rule 23:  Butter => Wine, Chicken; confidence: 0.75
Rule 24:  Wine => Butter, Chicken; confidence: 0.6
Rule 25:  Chicken => Butter, Wine; confidence: 0.6
Apriori execution time : 0.032
```



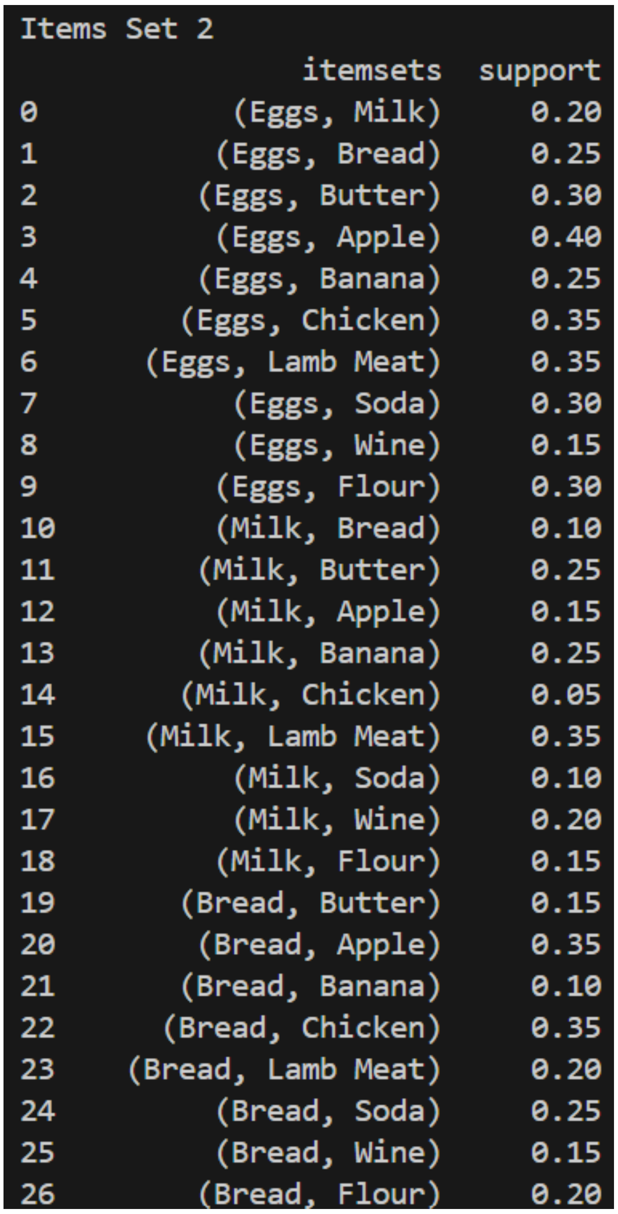
```
Executiong FP-growth Algorithm
Frequency Item Set using FP-Growth Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with non-bool types res
ult in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool typ
e
  warnings.warn(
Final Items Set:
    support                itemsets
0      0.55                  (Eggs)
1      0.50               (Chicken)
2      0.40                 (Flour)
3      0.40                (Banana)
4      0.35                  (Soda)
5      0.30                 (Bread)
6      0.65                  (Milk)
7      0.50                  (Wine)
8      0.40                (Butter)
9      0.30             (Lamb Meat)
10     0.30            (Eggs, Milk)
11     0.35         (Wine, Chicken)
12     0.30         (Milk, Chicken)
13     0.30       (Banana, Chicken)
14     0.30          (Banana, Wine)
15     0.30          (Banana, Soda)
16     0.30         (Chicken, Soda)
17     0.30            (Wine, Soda)
18     0.30            (Milk, Wine)
19     0.35       (Butter, Chicken)
20     0.35          (Butter, Wine)
21     0.30  (Butter, Wine, Chicken)
```



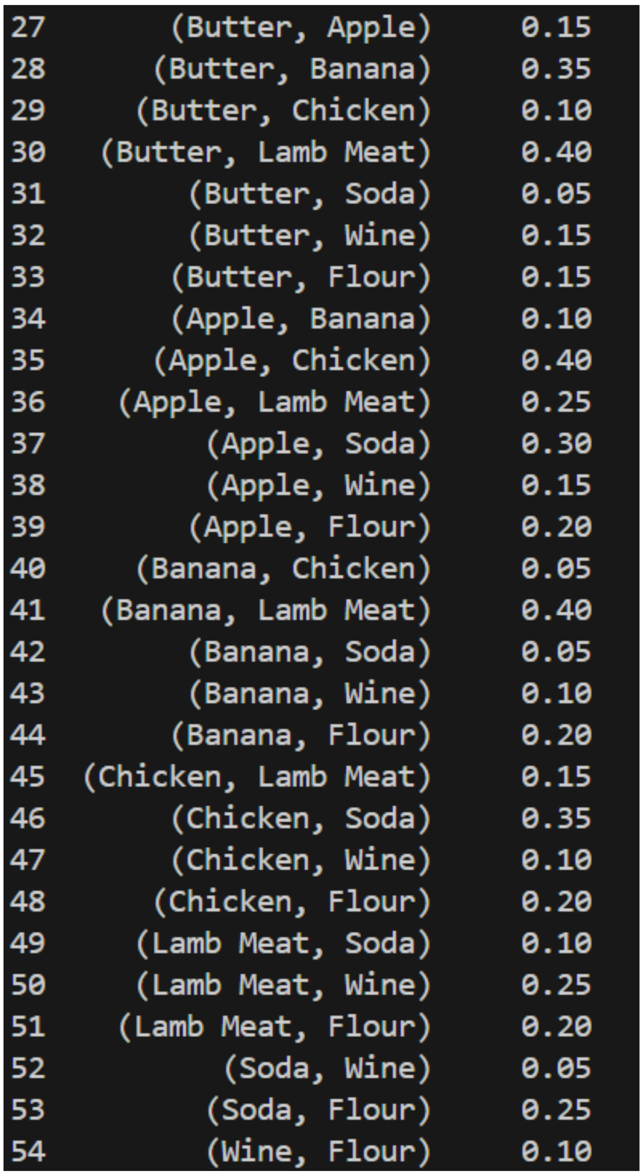
```
FP-Growth Rules:
Rule 1:  Eggs => Milk; confidence: 0.545
Rule 2:  Chicken => Milk; confidence: 0.6
Rule 3:  Wine => Milk; confidence: 0.6
Rule 4:  Butter => Chicken; confidence: 0.875
Rule 5:  Chicken => Butter; confidence: 0.7
Rule 6:  Butter => Wine; confidence: 0.875
Rule 7:  Wine => Butter; confidence: 0.7
Rule 8:  Banana => Chicken; confidence: 0.75
Rule 9:  Chicken => Banana; confidence: 0.6
Rule 10:  Banana => Soda; confidence: 0.75
Rule 11:  Soda => Banana; confidence: 0.857
Rule 12:  Banana => Wine; confidence: 0.75
Rule 13:  Wine => Banana; confidence: 0.6
Rule 14:  Chicken => Soda; confidence: 0.6
Rule 15:  Soda => Chicken; confidence: 0.857
Rule 16:  Wine => Chicken; confidence: 0.7
Rule 17:  Chicken => Wine; confidence: 0.7
Rule 18:  Wine => Soda; confidence: 0.6
Rule 19:  Soda => Wine; confidence: 0.857
Rule 20:  Butter, Wine => Chicken; confidence: 0.857
Rule 21:  Butter, Chicken => Wine; confidence: 0.857
Rule 22:  Wine, Chicken => Butter; confidence: 0.857
Rule 23:  Butter => Wine, Chicken; confidence: 0.75
Rule 24:  Wine => Butter, Chicken; confidence: 0.6
Rule 25:  Chicken => Butter, Wine; confidence: 0.6
FP-Growth Execution time : 0.035
Apriori is 9.0 times faster than brute force
FP-Growth is 8.0 times faster than brute force
FP-Growth is 0.0 time faster than Apriori
```



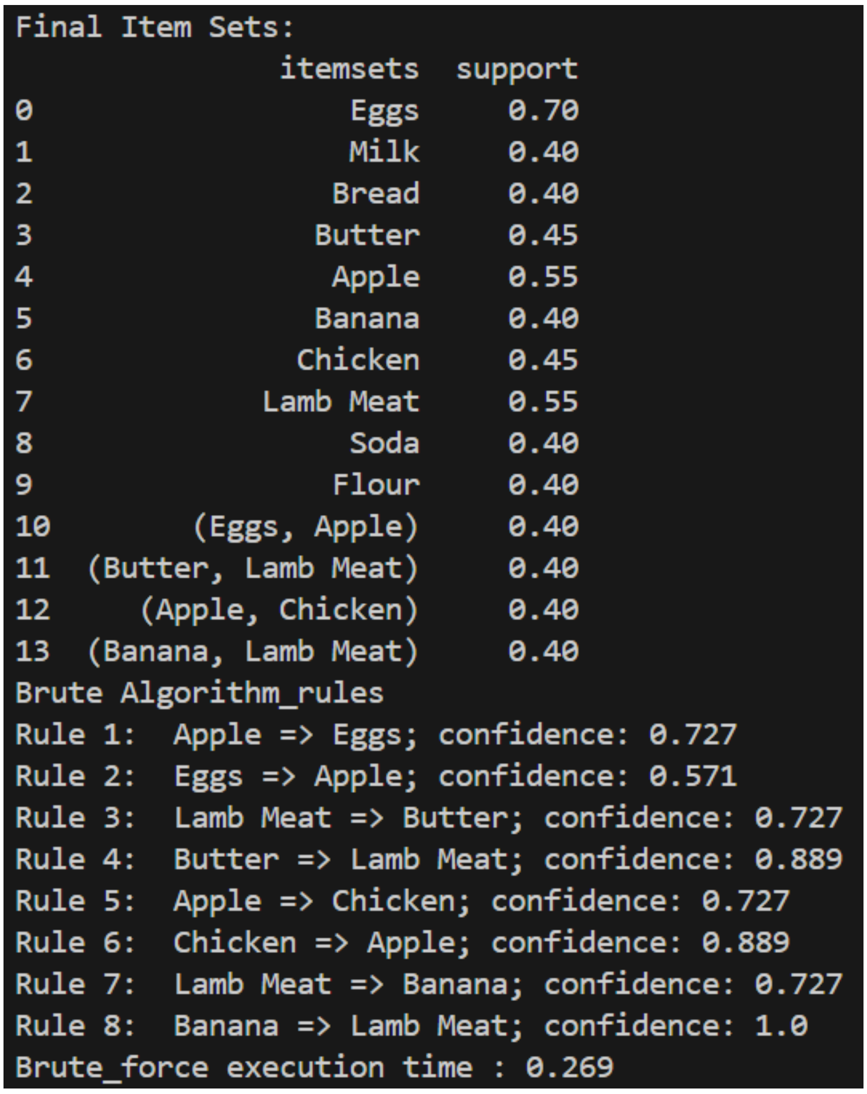
```
Processing Dataset 3: Data/Items3.csv
Enter min_threshold_support (in %): 40
Enter min_threshold_confidence  (in %): 40
Executing using Brute Force Algorithm
Item Set 1
     itemsets  support
0        Eggs     0.70
1        Milk     0.40
2       Bread     0.40
3      Butter     0.45
4       Apple     0.55
5      Banana     0.40
6     Chicken     0.45
7   Lamb Meat     0.55
8        Soda     0.40
9        Wine     0.25
10      Flour     0.40
```

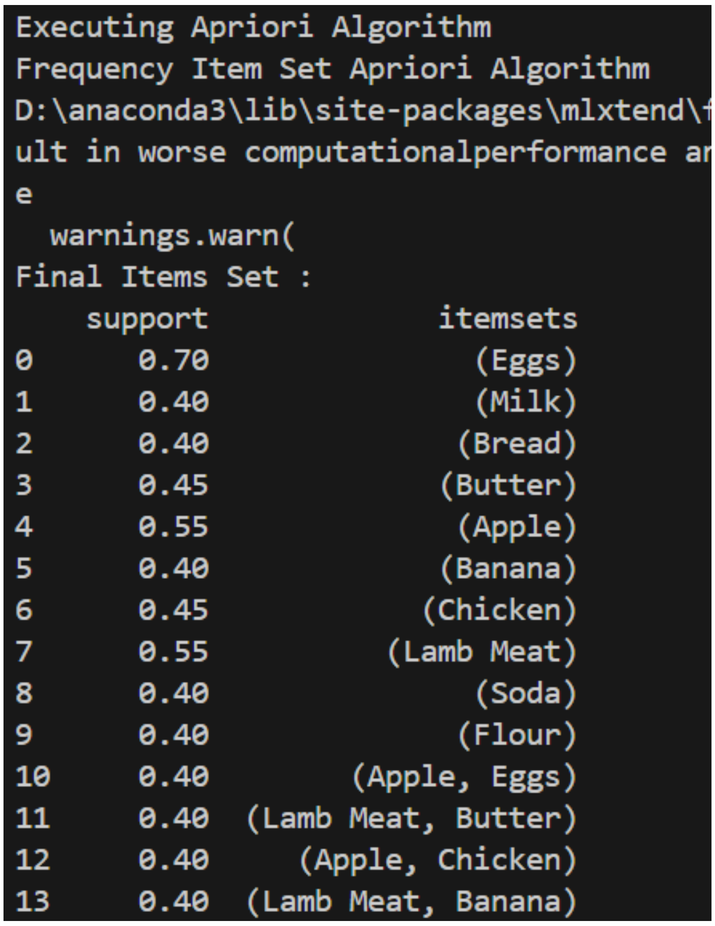


```
Items Set 2
                   itemsets  support
0              (Eggs, Milk)     0.20
1             (Eggs, Bread)     0.25
2            (Eggs, Butter)     0.30
3             (Eggs, Apple)     0.40
4            (Eggs, Banana)     0.25
5           (Eggs, Chicken)     0.35
6         (Eggs, Lamb Meat)     0.35
7              (Eggs, Soda)     0.30
8              (Eggs, Wine)     0.15
9             (Eggs, Flour)     0.30
10            (Milk, Bread)     0.10
11           (Milk, Butter)     0.25
12            (Milk, Apple)     0.15
13           (Milk, Banana)     0.25
14          (Milk, Chicken)     0.05
15        (Milk, Lamb Meat)     0.35
16             (Milk, Soda)     0.10
17             (Milk, Wine)     0.20
18            (Milk, Flour)     0.15
19          (Bread, Butter)     0.15
20           (Bread, Apple)     0.35
21          (Bread, Banana)     0.10
22         (Bread, Chicken)     0.35
23       (Bread, Lamb Meat)     0.20
24            (Bread, Soda)     0.25
25            (Bread, Wine)     0.15
26           (Bread, Flour)     0.20
```

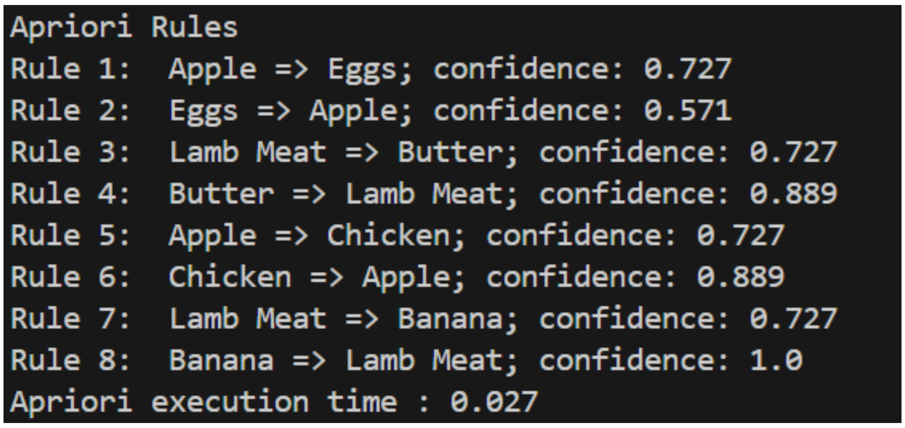


```
27          (Butter, Apple)      0.15
28         (Butter, Banana)      0.35
29        (Butter, Chicken)      0.10
30      (Butter, Lamb Meat)      0.40
31           (Butter, Soda)      0.05
32           (Butter, Wine)      0.15
33          (Butter, Flour)      0.15
34          (Apple, Banana)      0.10
35         (Apple, Chicken)      0.40
36       (Apple, Lamb Meat)      0.25
37            (Apple, Soda)      0.30
38            (Apple, Wine)      0.15
39           (Apple, Flour)      0.20
40        (Banana, Chicken)      0.05
41      (Banana, Lamb Meat)      0.40
42           (Banana, Soda)      0.05
43           (Banana, Wine)      0.10
44          (Banana, Flour)      0.20
45     (Chicken, Lamb Meat)      0.15
46          (Chicken, Soda)      0.35
47          (Chicken, Wine)      0.10
48         (Chicken, Flour)      0.20
49        (Lamb Meat, Soda)      0.10
50        (Lamb Meat, Wine)      0.25
51       (Lamb Meat, Flour)      0.20
52             (Soda, Wine)      0.05
53            (Soda, Flour)      0.25
54            (Wine, Flour)      0.10
```

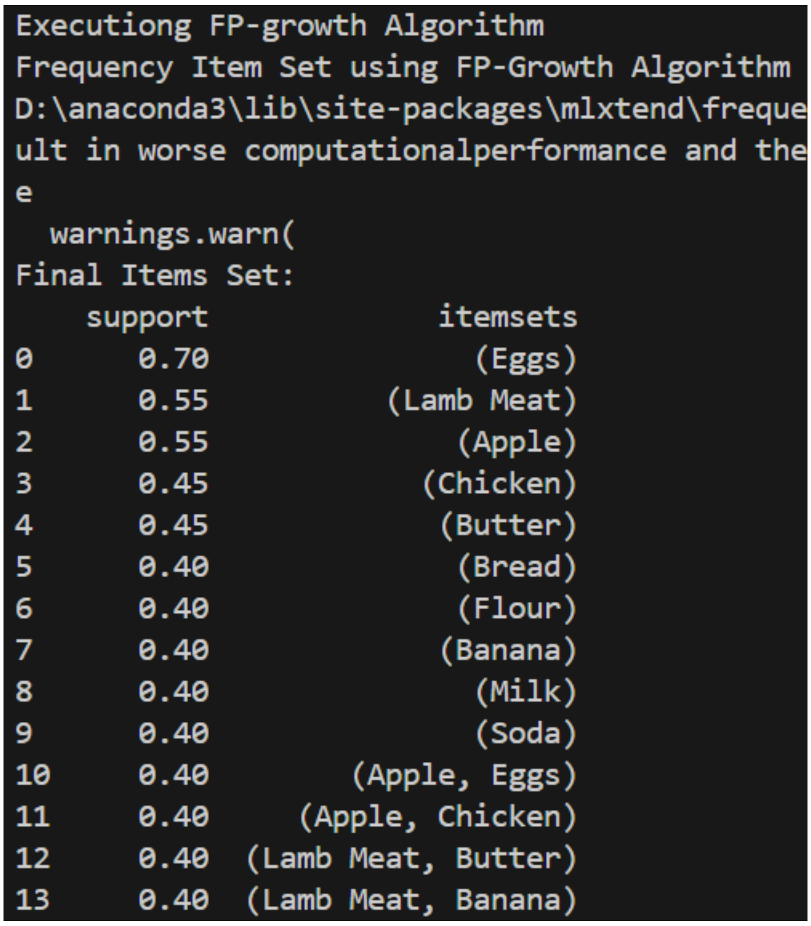

```
Final Item Sets:
                  itemsets  support
0                     Eggs     0.70
1                     Milk     0.40
2                    Bread     0.40
3                   Butter     0.45
4                    Apple     0.55
5                   Banana     0.40
6                  Chicken     0.45
7                Lamb Meat     0.55
8                     Soda     0.40
9                    Flour     0.40
10           (Eggs, Apple)     0.40
11   (Butter, Lamb Meat)     0.40
12        (Apple, Chicken)     0.40
13   (Banana, Lamb Meat)     0.40
Brute Algorithm_rules
Rule 1:  Apple => Eggs; confidence: 0.727
Rule 2:  Eggs => Apple; confidence: 0.571
Rule 3:  Lamb Meat => Butter; confidence: 0.727
Rule 4:  Butter => Lamb Meat; confidence: 0.889
Rule 5:  Apple => Chicken; confidence: 0.727
Rule 6:  Chicken => Apple; confidence: 0.889
Rule 7:  Lamb Meat => Banana; confidence: 0.727
Rule 8:  Banana => Lamb Meat; confidence: 1.0
Brute_force execution time : 0.269
```

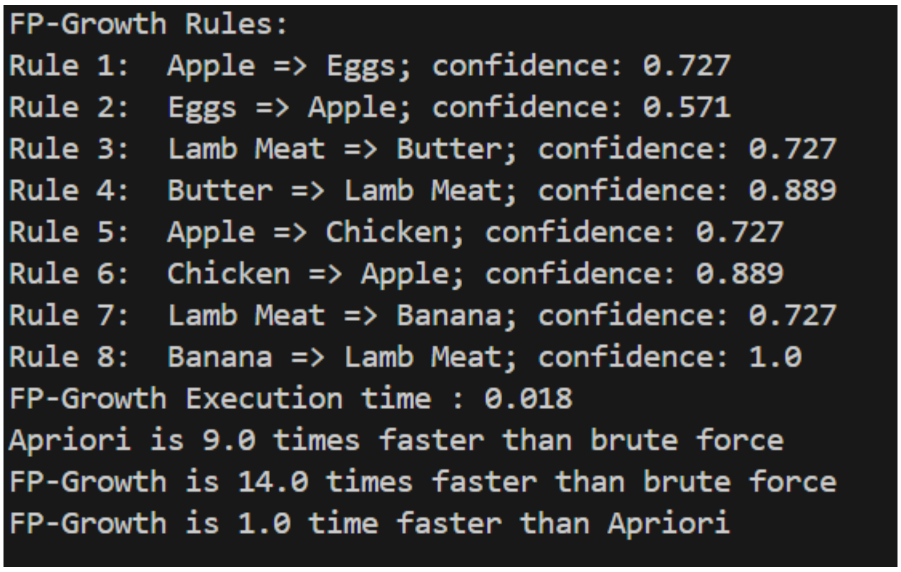


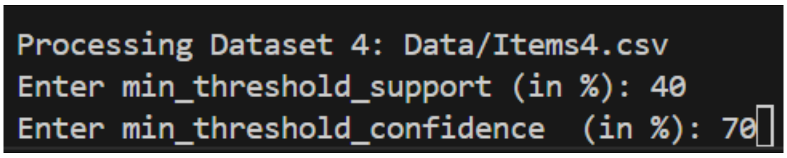
```
Executing Apriori Algorithm
Frequency Item Set Apriori Algorithm
D:\anaconda3\lib\site-packages\mlxtend\f
ult in worse computationalperformance ar
e
  warnings.warn(
Final Items Set :
    support              itemsets
0      0.70                (Eggs)
1      0.40                (Milk)
2      0.40               (Bread)
3      0.45              (Butter)
4      0.55               (Apple)
5      0.40              (Banana)
6      0.45             (Chicken)
7      0.55           (Lamb Meat)
8      0.40                (Soda)
9      0.40               (Flour)
10     0.40         (Apple, Eggs)
11     0.40  (Lamb Meat, Butter)
12     0.40      (Apple, Chicken)
13     0.40  (Lamb Meat, Banana)
```

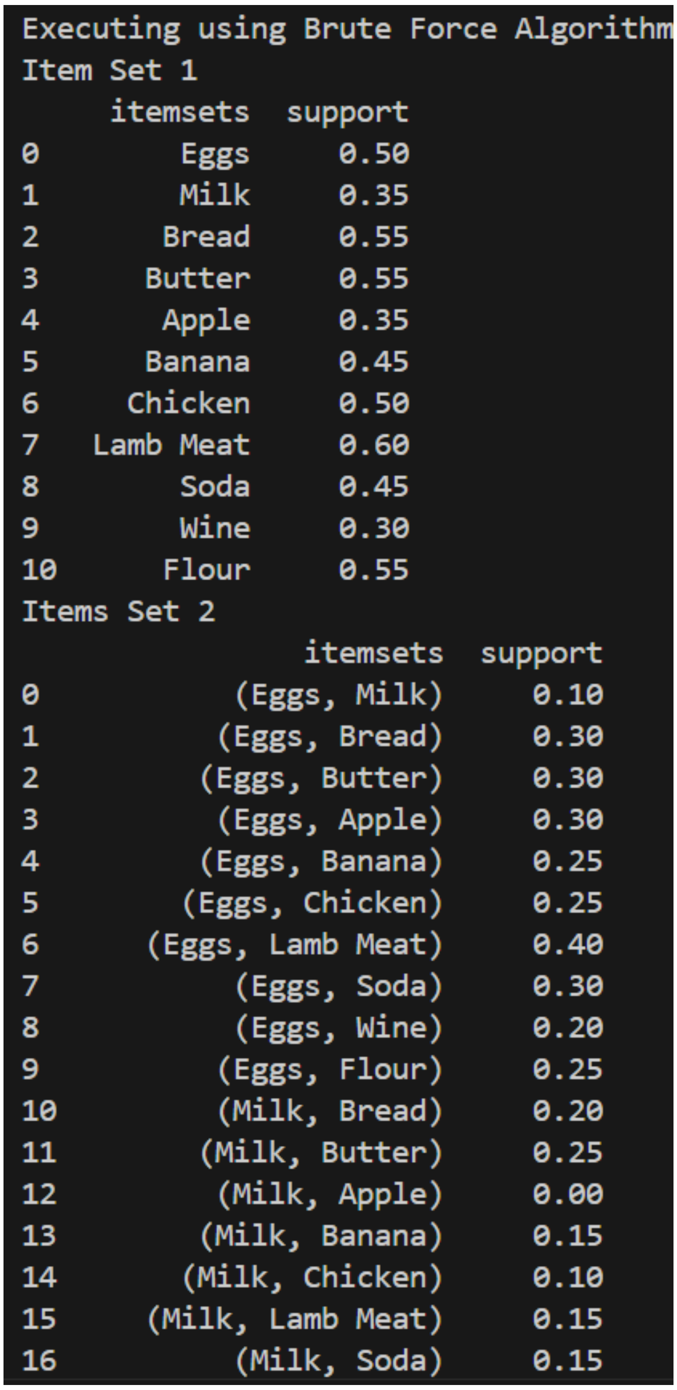


```
Apriori Rules
Rule 1:  Apple => Eggs; confidence: 0.727
Rule 2:  Eggs => Apple; confidence: 0.571
Rule 3:  Lamb Meat => Butter; confidence: 0.727
Rule 4:  Butter => Lamb Meat; confidence: 0.889
Rule 5:  Apple => Chicken; confidence: 0.727
Rule 6:  Chicken => Apple; confidence: 0.889
Rule 7:  Lamb Meat => Banana; confidence: 0.727
Rule 8:  Banana => Lamb Meat; confidence: 1.0
Apriori execution time : 0.027
```

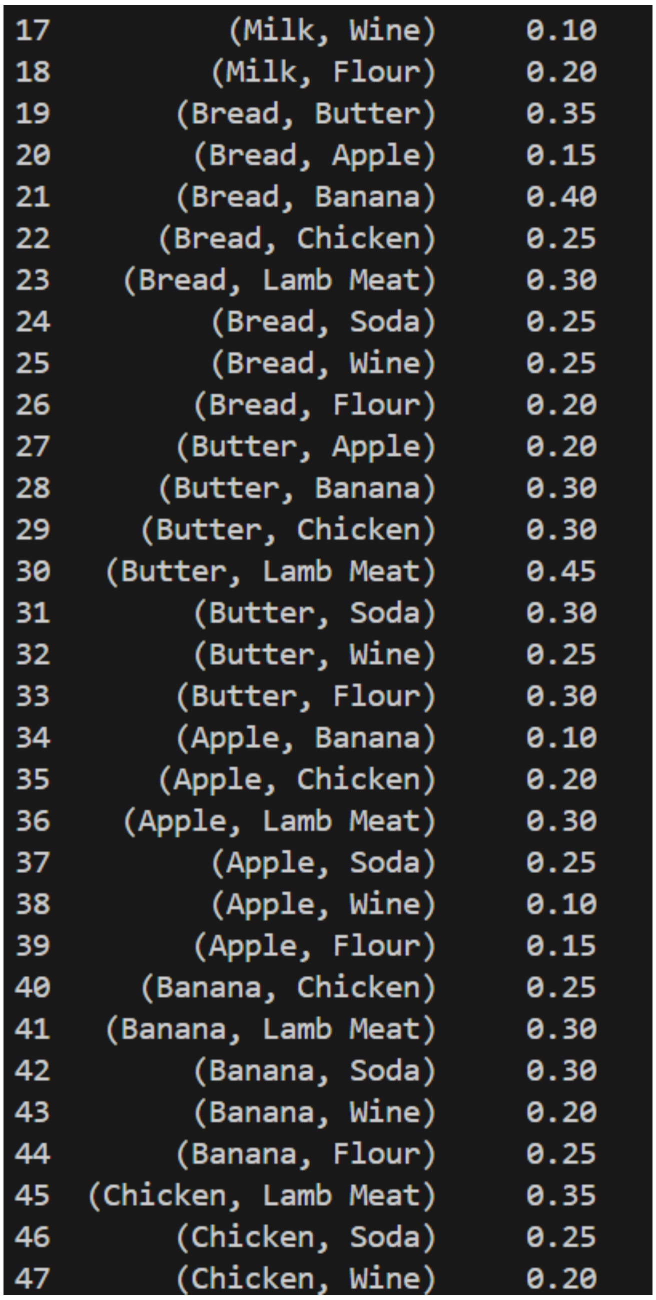

```
Executiong FP-growth Algorithm
Frequency Item Set using FP-Growth Algorithm
D:\anaconda3\lib\site-packages\mlxtend\freque
ult in worse computationalperformance and the
e
  warnings.warn(
Final Items Set:
    support              itemsets
0      0.70                (Eggs)
1      0.55           (Lamb Meat)
2      0.55               (Apple)
3      0.45             (Chicken)
4      0.45              (Butter)
5      0.40               (Bread)
6      0.40               (Flour)
7      0.40              (Banana)
8      0.40                (Milk)
9      0.40                (Soda)
10     0.40         (Apple, Eggs)
11     0.40      (Apple, Chicken)
12     0.40  (Lamb Meat, Butter)
13     0.40  (Lamb Meat, Banana)
```


```
FP-Growth Rules:
Rule 1:  Apple => Eggs; confidence: 0.727
Rule 2:  Eggs => Apple; confidence: 0.571
Rule 3:  Lamb Meat => Butter; confidence: 0.727
Rule 4:  Butter => Lamb Meat; confidence: 0.889
Rule 5:  Apple => Chicken; confidence: 0.727
Rule 6:  Chicken => Apple; confidence: 0.889
Rule 7:  Lamb Meat => Banana; confidence: 0.727
Rule 8:  Banana => Lamb Meat; confidence: 1.0
FP-Growth Execution time : 0.018
Apriori is 9.0 times faster than brute force
FP-Growth is 14.0 times faster than brute force
FP-Growth is 1.0 time faster than Apriori
```

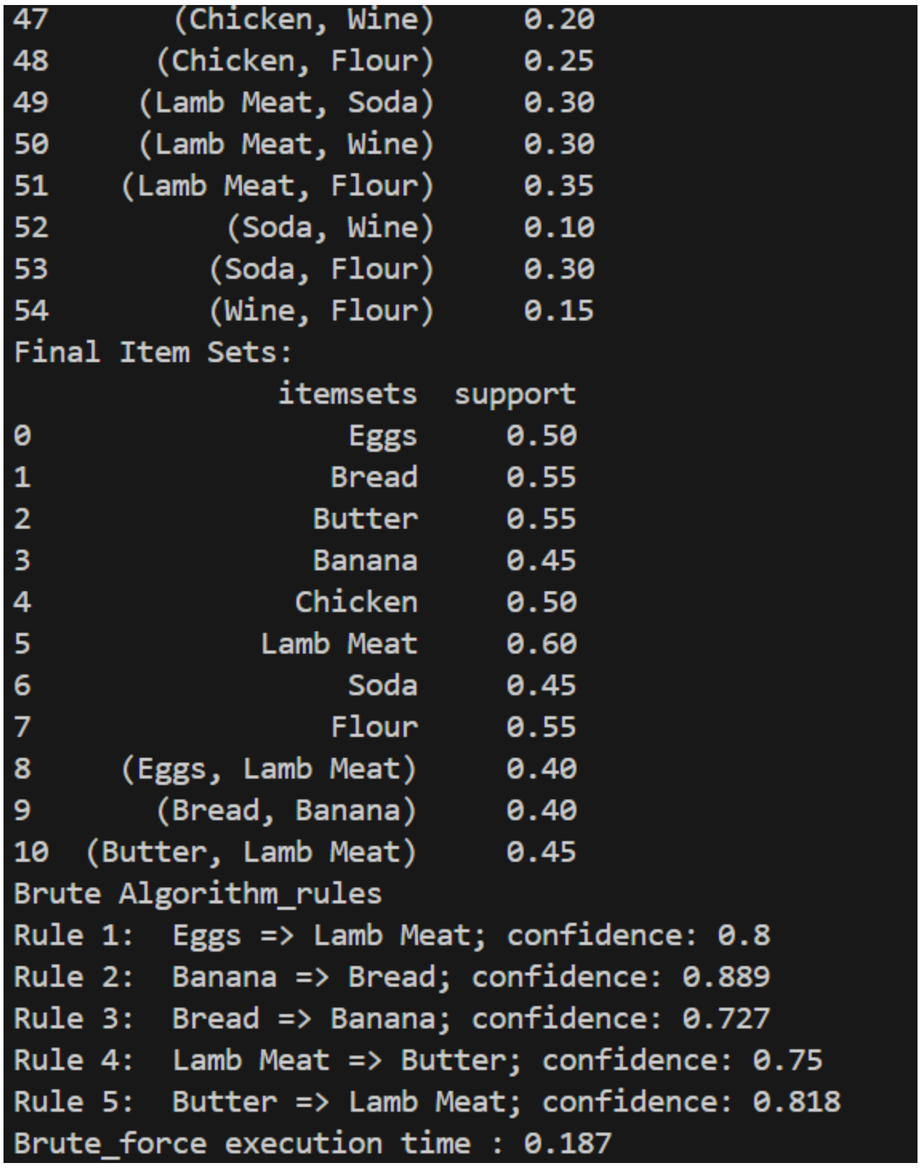

```
Processing Dataset 4: Data/Items4.csv
Enter min_threshold_support (in %): 40
Enter min_threshold_confidence  (in %): 70
```

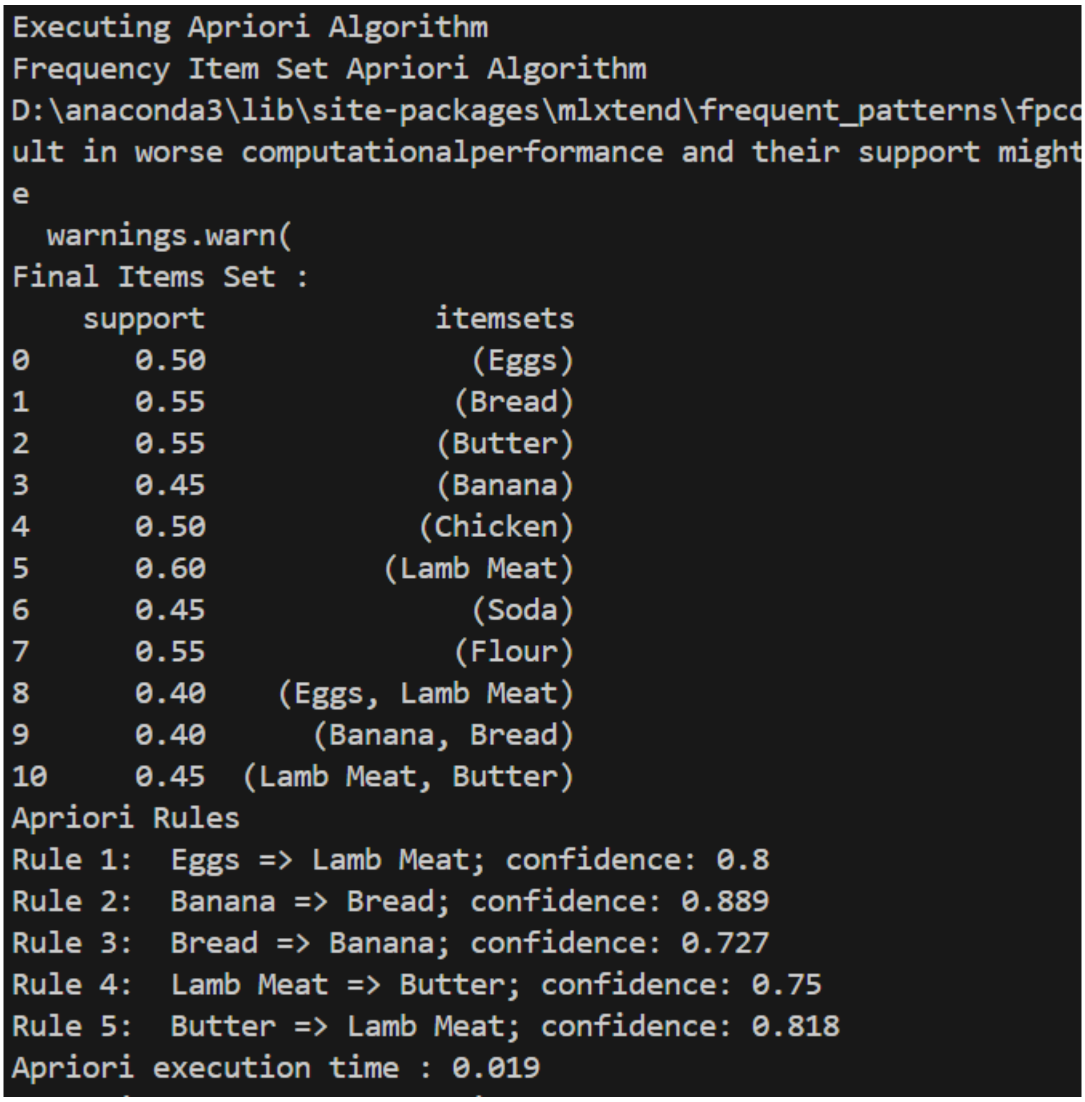


```
Executing using Brute Force Algorithm
Item Set 1
     itemsets  support
0        Eggs     0.50
1        Milk     0.35
2       Bread     0.55
3      Butter     0.55
4       Apple     0.35
5      Banana     0.45
6     Chicken     0.50
7   Lamb Meat     0.60
8        Soda     0.45
9        Wine     0.30
10      Flour     0.55
Items Set 2
                itemsets  support
0           (Eggs, Milk)     0.10
1          (Eggs, Bread)     0.30
2         (Eggs, Butter)     0.30
3          (Eggs, Apple)     0.30
4         (Eggs, Banana)     0.25
5        (Eggs, Chicken)     0.25
6      (Eggs, Lamb Meat)     0.40
7           (Eggs, Soda)     0.30
8           (Eggs, Wine)     0.20
9          (Eggs, Flour)     0.25
10         (Milk, Bread)     0.20
11        (Milk, Butter)     0.25
12         (Milk, Apple)     0.00
13        (Milk, Banana)     0.15
14       (Milk, Chicken)     0.10
15     (Milk, Lamb Meat)     0.15
16          (Milk, Soda)     0.15
```

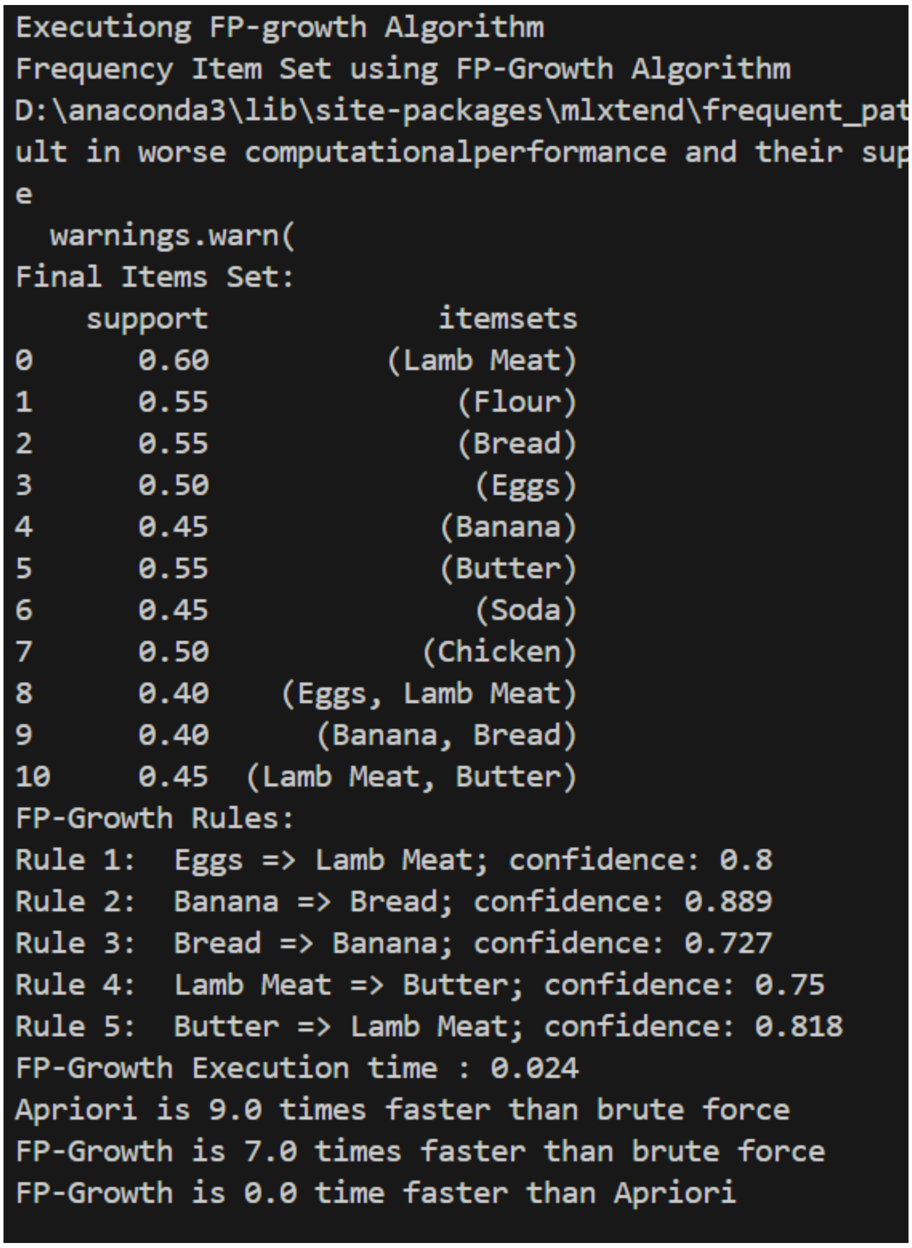


```
17            (Milk, Wine)      0.10
18           (Milk, Flour)      0.20
19         (Bread, Butter)      0.35
20          (Bread, Apple)      0.15
21         (Bread, Banana)      0.40
22        (Bread, Chicken)      0.25
23      (Bread, Lamb Meat)      0.30
24           (Bread, Soda)      0.25
25           (Bread, Wine)      0.25
26          (Bread, Flour)      0.20
27         (Butter, Apple)      0.20
28        (Butter, Banana)      0.30
29       (Butter, Chicken)      0.30
30     (Butter, Lamb Meat)      0.45
31          (Butter, Soda)      0.30
32          (Butter, Wine)      0.25
33         (Butter, Flour)      0.30
34         (Apple, Banana)      0.10
35        (Apple, Chicken)      0.20
36      (Apple, Lamb Meat)      0.30
37           (Apple, Soda)      0.25
38           (Apple, Wine)      0.10
39          (Apple, Flour)      0.15
40       (Banana, Chicken)      0.25
41     (Banana, Lamb Meat)      0.30
42          (Banana, Soda)      0.30
43          (Banana, Wine)      0.20
44         (Banana, Flour)      0.25
45    (Chicken, Lamb Meat)      0.35
46         (Chicken, Soda)      0.25
47         (Chicken, Wine)      0.20
```

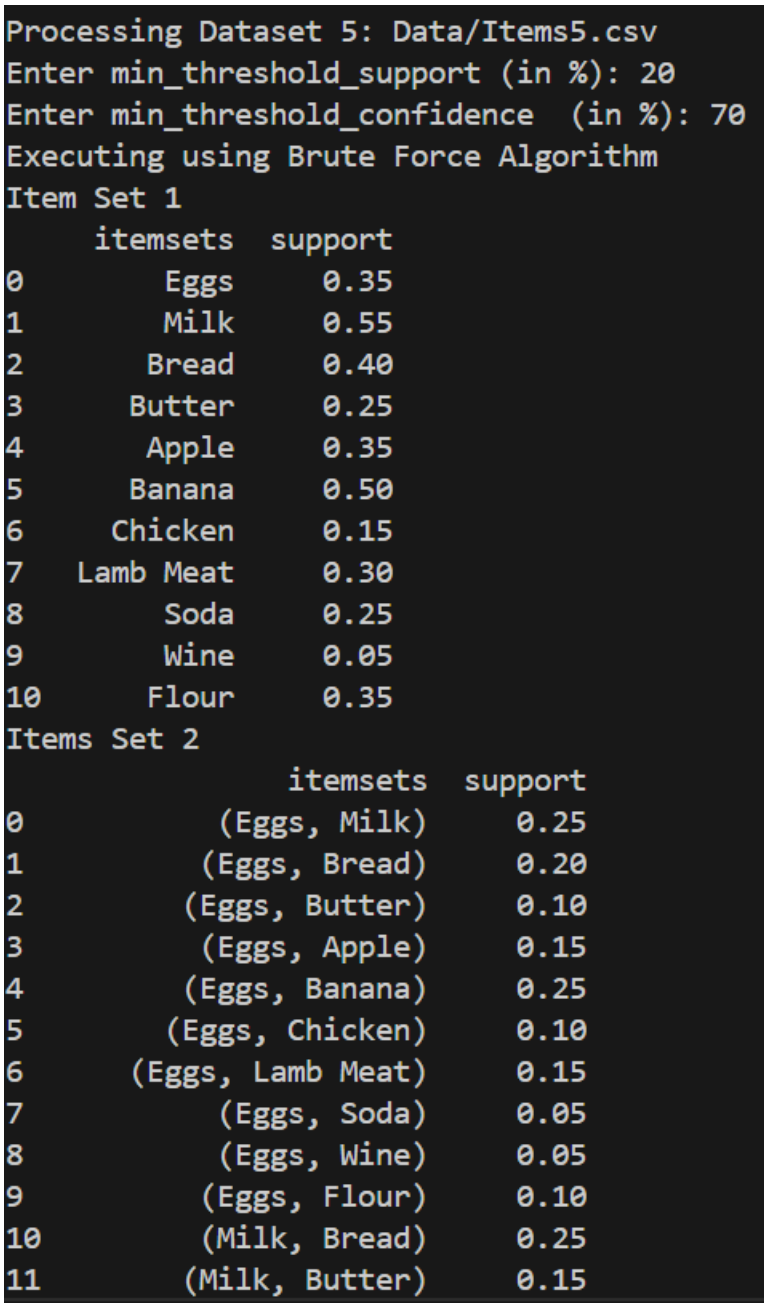

```
47       (Chicken, Wine)     0.20
48      (Chicken, Flour)     0.25
49     (Lamb Meat, Soda)     0.30
50     (Lamb Meat, Wine)     0.30
51    (Lamb Meat, Flour)     0.35
52          (Soda, Wine)     0.10
53         (Soda, Flour)     0.30
54         (Wine, Flour)     0.15
Final Item Sets:
               itemsets  support
0                  Eggs     0.50
1                 Bread     0.55
2                Butter     0.55
3                Banana     0.45
4               Chicken     0.50
5             Lamb Meat     0.60
6                  Soda     0.45
7                 Flour     0.55
8     (Eggs, Lamb Meat)     0.40
9       (Bread, Banana)     0.40
10  (Butter, Lamb Meat)     0.45
Brute Algorithm_rules
Rule 1:  Eggs => Lamb Meat; confidence: 0.8
Rule 2:  Banana => Bread; confidence: 0.889
Rule 3:  Bread => Banana; confidence: 0.727
Rule 4:  Lamb Meat => Butter; confidence: 0.75
Rule 5:  Butter => Lamb Meat; confidence: 0.818
Brute_force execution time : 0.187
```



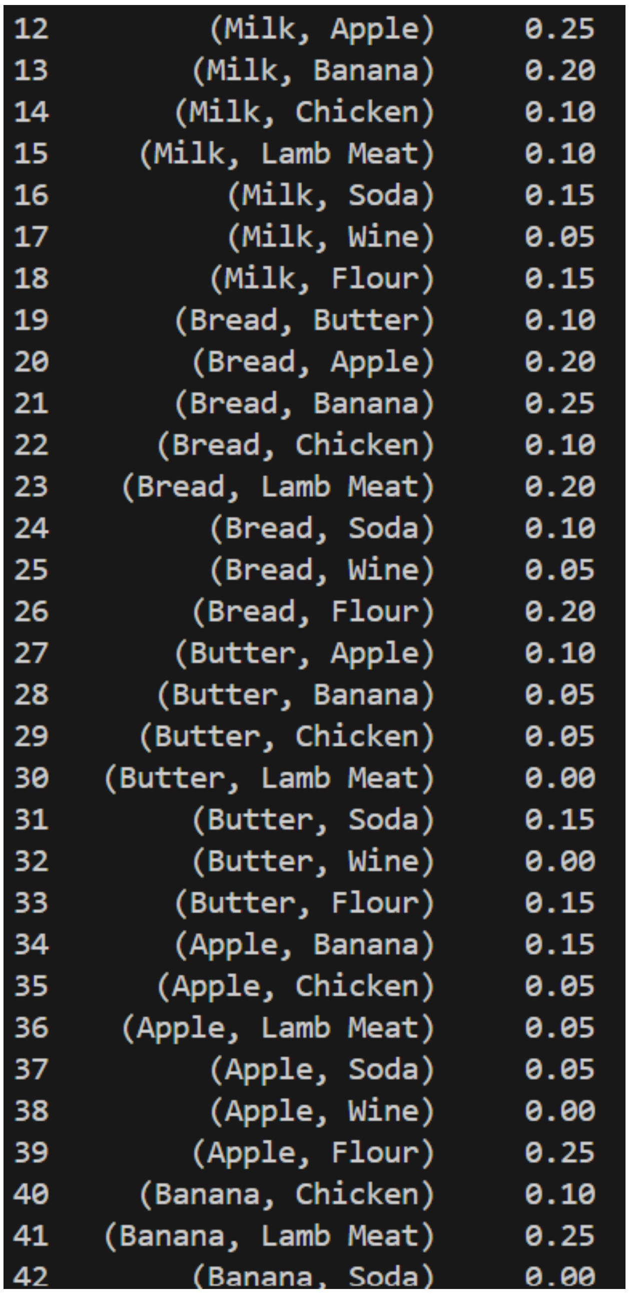
```
Executing Apriori Algorithm
Frequency Item Set Apriori Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpco
ult in worse computationalperformance and their support might
e
  warnings.warn(
Final Items Set :
    support             itemsets
0      0.50               (Eggs)
1      0.55              (Bread)
2      0.55             (Butter)
3      0.45             (Banana)
4      0.50            (Chicken)
5      0.60          (Lamb Meat)
6      0.45               (Soda)
7      0.55              (Flour)
8      0.40    (Eggs, Lamb Meat)
9      0.40      (Banana, Bread)
10     0.45  (Lamb Meat, Butter)
Apriori Rules
Rule 1:  Eggs => Lamb Meat; confidence: 0.8
Rule 2:  Banana => Bread; confidence: 0.889
Rule 3:  Bread => Banana; confidence: 0.727
Rule 4:  Lamb Meat => Butter; confidence: 0.75
Rule 5:  Butter => Lamb Meat; confidence: 0.818
Apriori execution time : 0.019
```



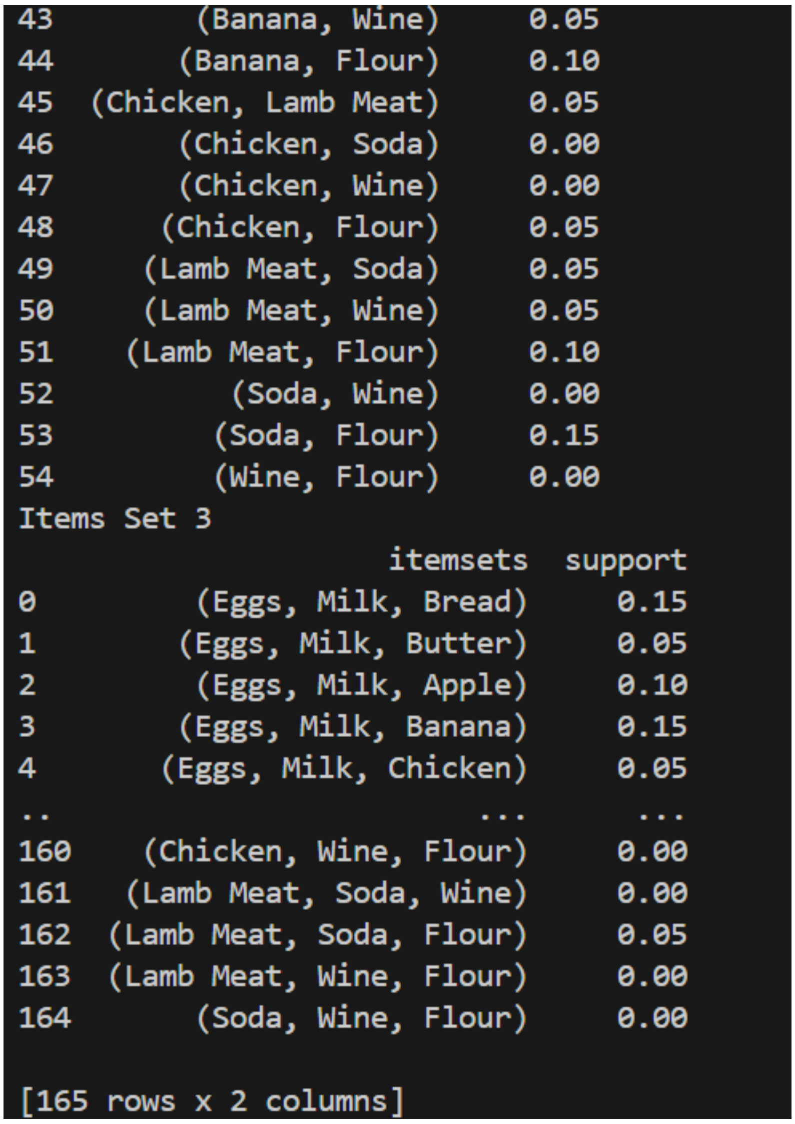
```
Executiong FP-growth Algorithm
Frequency Item Set using FP-Growth Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_pat
ult in worse computationalperformance and their sup
e
  warnings.warn(
Final Items Set:
    support              itemsets
0      0.60           (Lamb Meat)
1      0.55               (Flour)
2      0.55               (Bread)
3      0.50                (Eggs)
4      0.45              (Banana)
5      0.55              (Butter)
6      0.45                (Soda)
7      0.50             (Chicken)
8      0.40     (Eggs, Lamb Meat)
9      0.40       (Banana, Bread)
10     0.45   (Lamb Meat, Butter)
FP-Growth Rules:
Rule 1:  Eggs => Lamb Meat; confidence: 0.8
Rule 2:  Banana => Bread; confidence: 0.889
Rule 3:  Bread => Banana; confidence: 0.727
Rule 4:  Lamb Meat => Butter; confidence: 0.75
Rule 5:  Butter => Lamb Meat; confidence: 0.818
FP-Growth Execution time : 0.024
Apriori is 9.0 times faster than brute force
FP-Growth is 7.0 times faster than brute force
FP-Growth is 0.0 time faster than Apriori
```



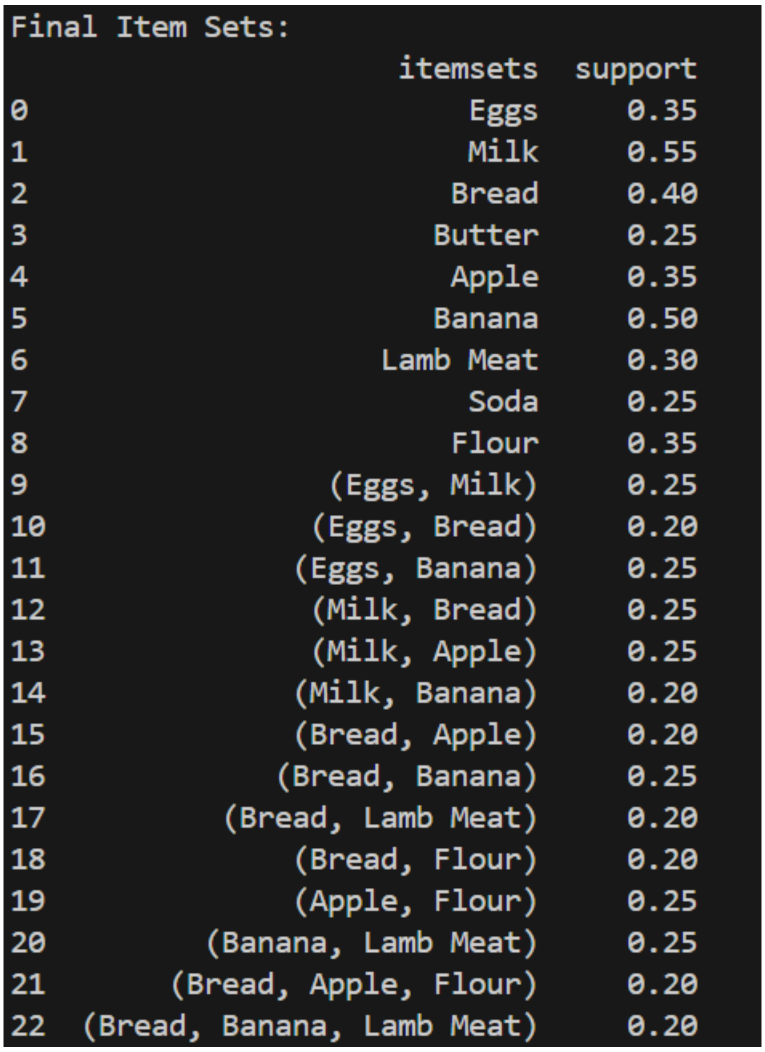
```
Processing Dataset 5: Data/Items5.csv
Enter min_threshold_support (in %): 20
Enter min_threshold_confidence  (in %): 70
Executing using Brute Force Algorithm
Item Set 1
     itemsets  support
0        Eggs     0.35
1        Milk     0.55
2       Bread     0.40
3      Butter     0.25
4       Apple     0.35
5      Banana     0.50
6     Chicken     0.15
7   Lamb Meat     0.30
8        Soda     0.25
9        Wine     0.05
10      Flour     0.35
Items Set 2
                itemsets  support
0           (Eggs, Milk)     0.25
1          (Eggs, Bread)     0.20
2         (Eggs, Butter)     0.10
3          (Eggs, Apple)     0.15
4         (Eggs, Banana)     0.25
5        (Eggs, Chicken)     0.10
6      (Eggs, Lamb Meat)     0.15
7           (Eggs, Soda)     0.05
8           (Eggs, Wine)     0.05
9          (Eggs, Flour)     0.10
10         (Milk, Bread)     0.25
11        (Milk, Butter)     0.15
```



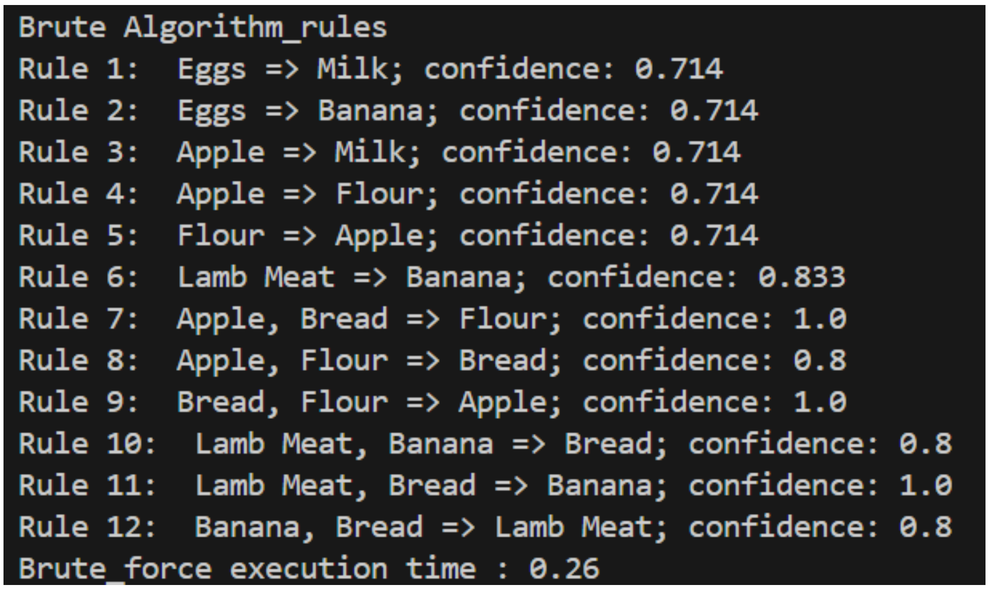
```
12          (Milk, Apple)      0.25
13         (Milk, Banana)      0.20
14        (Milk, Chicken)      0.10
15      (Milk, Lamb Meat)      0.10
16           (Milk, Soda)      0.15
17           (Milk, Wine)      0.05
18          (Milk, Flour)      0.15
19        (Bread, Butter)      0.10
20         (Bread, Apple)      0.20
21        (Bread, Banana)      0.25
22       (Bread, Chicken)      0.10
23     (Bread, Lamb Meat)      0.20
24          (Bread, Soda)      0.10
25          (Bread, Wine)      0.05
26         (Bread, Flour)      0.20
27        (Butter, Apple)      0.10
28       (Butter, Banana)      0.05
29      (Butter, Chicken)      0.05
30    (Butter, Lamb Meat)      0.00
31         (Butter, Soda)      0.15
32         (Butter, Wine)      0.00
33        (Butter, Flour)      0.15
34        (Apple, Banana)      0.15
35       (Apple, Chicken)      0.05
36     (Apple, Lamb Meat)      0.05
37          (Apple, Soda)      0.05
38          (Apple, Wine)      0.00
39         (Apple, Flour)      0.25
40      (Banana, Chicken)      0.10
41    (Banana, Lamb Meat)      0.25
42         (Banana, Soda)      0.00
```



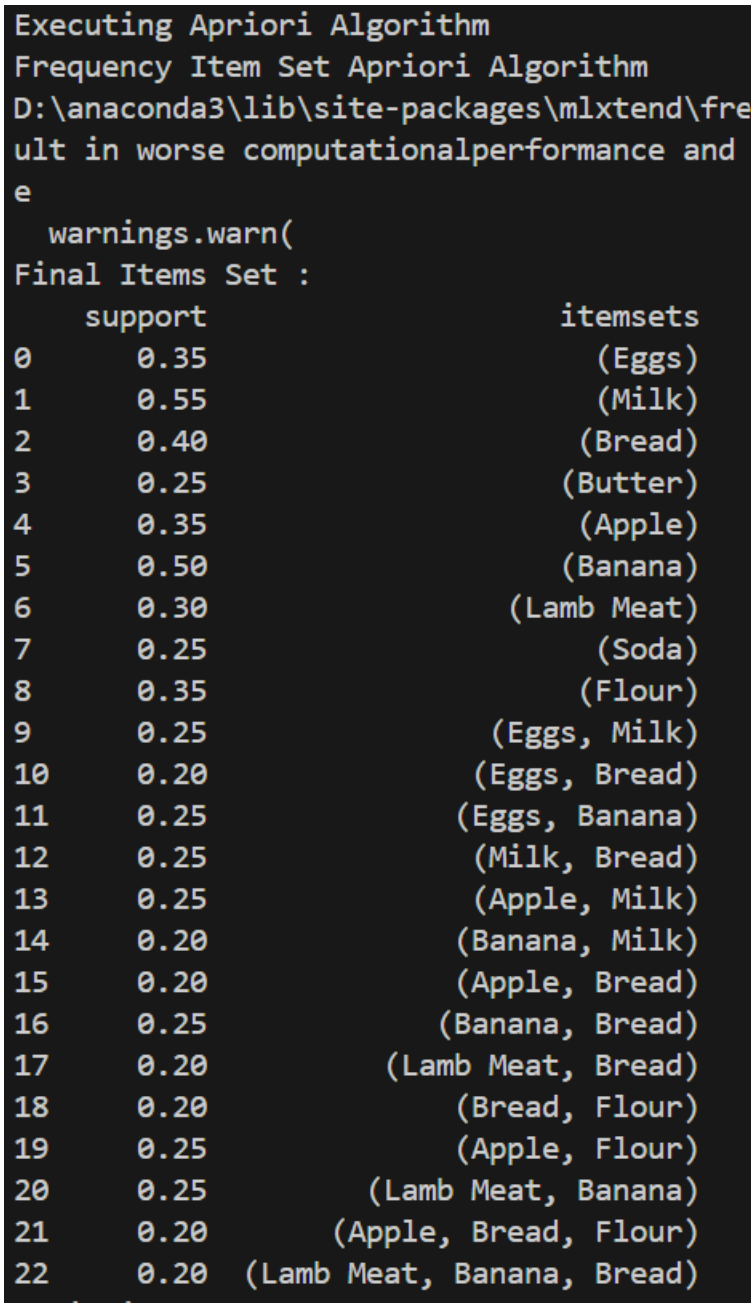
```
43         (Banana, Wine)     0.05
44        (Banana, Flour)     0.10
45   (Chicken, Lamb Meat)     0.05
46        (Chicken, Soda)     0.00
47        (Chicken, Wine)     0.00
48       (Chicken, Flour)     0.05
49      (Lamb Meat, Soda)     0.05
50      (Lamb Meat, Wine)     0.05
51     (Lamb Meat, Flour)     0.10
52           (Soda, Wine)     0.00
53          (Soda, Flour)     0.15
54          (Wine, Flour)     0.00
Items Set 3
                     itemsets  support
0         (Eggs, Milk, Bread)     0.15
1        (Eggs, Milk, Butter)     0.05
2         (Eggs, Milk, Apple)     0.10
3        (Eggs, Milk, Banana)     0.15
4       (Eggs, Milk, Chicken)     0.05
..                        ...      ...
160    (Chicken, Wine, Flour)     0.00
161   (Lamb Meat, Soda, Wine)     0.00
162  (Lamb Meat, Soda, Flour)     0.05
163  (Lamb Meat, Wine, Flour)     0.00
164        (Soda, Wine, Flour)     0.00

[165 rows x 2 columns]
```



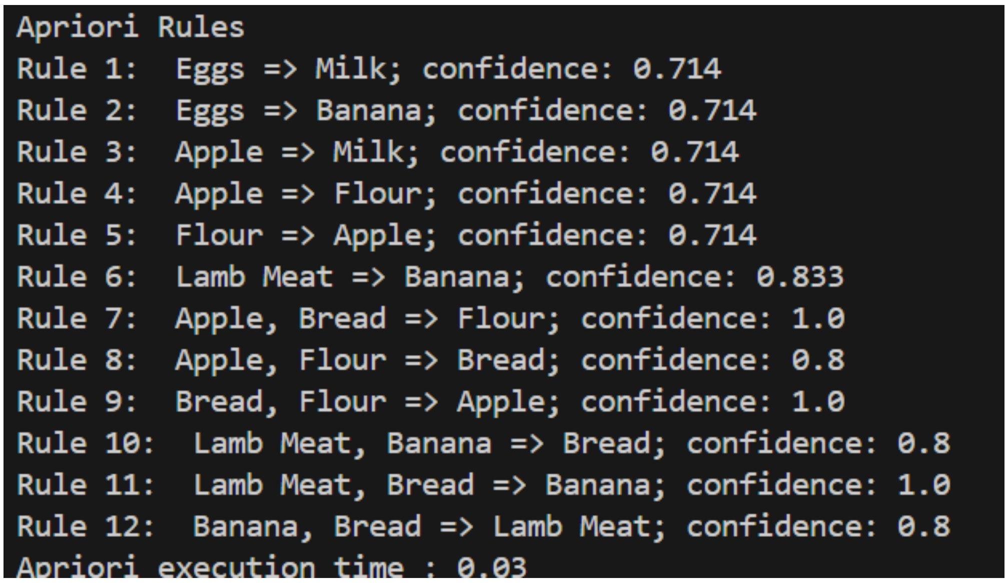
```
Final Item Sets:
                         itemsets  support
0                            Eggs     0.35
1                            Milk     0.55
2                           Bread     0.40
3                          Butter     0.25
4                           Apple     0.35
5                          Banana     0.50
6                       Lamb Meat     0.30
7                            Soda     0.25
8                           Flour     0.35
9                   (Eggs, Milk)     0.25
10                 (Eggs, Bread)     0.20
11                (Eggs, Banana)     0.25
12                 (Milk, Bread)     0.25
13                 (Milk, Apple)     0.25
14                (Milk, Banana)     0.20
15                (Bread, Apple)     0.20
16               (Bread, Banana)     0.25
17            (Bread, Lamb Meat)     0.20
18                (Bread, Flour)     0.20
19                (Apple, Flour)     0.25
20           (Banana, Lamb Meat)     0.25
21         (Bread, Apple, Flour)     0.20
22  (Bread, Banana, Lamb Meat)     0.20
```



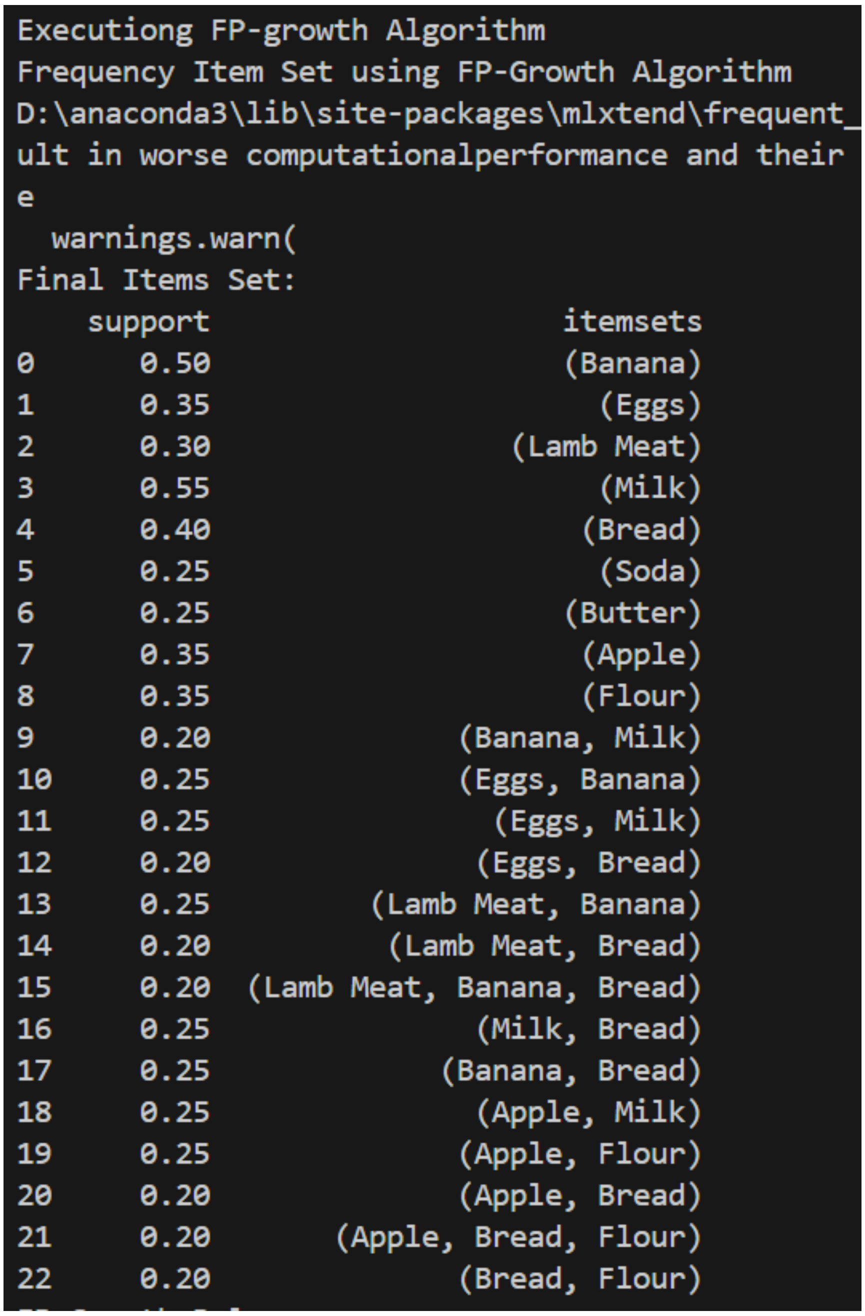
```
Brute Algorithm_rules
Rule 1:  Eggs => Milk; confidence: 0.714
Rule 2:  Eggs => Banana; confidence: 0.714
Rule 3:  Apple => Milk; confidence: 0.714
Rule 4:  Apple => Flour; confidence: 0.714
Rule 5:  Flour => Apple; confidence: 0.714
Rule 6:  Lamb Meat => Banana; confidence: 0.833
Rule 7:  Apple, Bread => Flour; confidence: 1.0
Rule 8:  Apple, Flour => Bread; confidence: 0.8
Rule 9:  Bread, Flour => Apple; confidence: 1.0
Rule 10:  Lamb Meat, Banana => Bread; confidence: 0.8
Rule 11:  Lamb Meat, Bread => Banana; confidence: 1.0
Rule 12:  Banana, Bread => Lamb Meat; confidence: 0.8
Brute_force execution time : 0.26
```



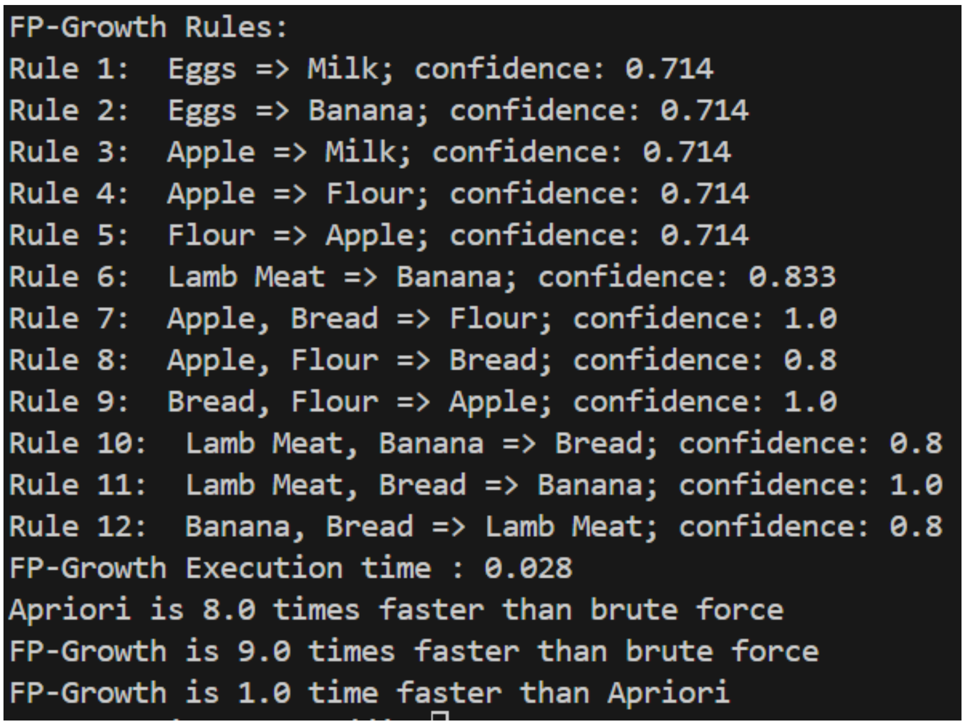
```
Executing Apriori Algorithm
Frequency Item Set Apriori Algorithm
D:\anaconda3\lib\site-packages\mlxtend\fre
ult in worse computationalperformance and
e
  warnings.warn(
Final Items Set :
    support                      itemsets
0      0.35                        (Eggs)
1      0.55                        (Milk)
2      0.40                       (Bread)
3      0.25                      (Butter)
4      0.35                       (Apple)
5      0.50                      (Banana)
6      0.30                   (Lamb Meat)
7      0.25                        (Soda)
8      0.35                       (Flour)
9      0.25                  (Eggs, Milk)
10     0.20                 (Eggs, Bread)
11     0.25                (Eggs, Banana)
12     0.25                 (Milk, Bread)
13     0.25                 (Apple, Milk)
14     0.20                (Banana, Milk)
15     0.20                (Apple, Bread)
16     0.25               (Banana, Bread)
17     0.20            (Lamb Meat, Bread)
18     0.20                (Bread, Flour)
19     0.25                (Apple, Flour)
20     0.25           (Lamb Meat, Banana)
21     0.20         (Apple, Bread, Flour)
22     0.20  (Lamb Meat, Banana, Bread)
```



```
Apriori Rules
Rule 1:  Eggs => Milk; confidence: 0.714
Rule 2:  Eggs => Banana; confidence: 0.714
Rule 3:  Apple => Milk; confidence: 0.714
Rule 4:  Apple => Flour; confidence: 0.714
Rule 5:  Flour => Apple; confidence: 0.714
Rule 6:  Lamb Meat => Banana; confidence: 0.833
Rule 7:  Apple, Bread => Flour; confidence: 1.0
Rule 8:  Apple, Flour => Bread; confidence: 0.8
Rule 9:  Bread, Flour => Apple; confidence: 1.0
Rule 10:  Lamb Meat, Banana => Bread; confidence: 0.8
Rule 11:  Lamb Meat, Bread => Banana; confidence: 1.0
Rule 12:  Banana, Bread => Lamb Meat; confidence: 0.8
Apriori execution time : 0.03
```



```
Executiong FP-growth Algorithm
Frequency Item Set using FP-Growth Algorithm
D:\anaconda3\lib\site-packages\mlxtend\frequent_
ult in worse computationalperformance and their
e
  warnings.warn(
Final Items Set:
    support                       itemsets
0      0.50                       (Banana)
1      0.35                         (Eggs)
2      0.30                    (Lamb Meat)
3      0.55                         (Milk)
4      0.40                        (Bread)
5      0.25                         (Soda)
6      0.25                       (Butter)
7      0.35                        (Apple)
8      0.35                        (Flour)
9      0.20                 (Banana, Milk)
10     0.20                 (Eggs, Banana)
11     0.25                   (Eggs, Milk)
12     0.20                  (Eggs, Bread)
13     0.25            (Lamb Meat, Banana)
14     0.20             (Lamb Meat, Bread)
15     0.20     (Lamb Meat, Banana, Bread)
16     0.25                  (Milk, Bread)
17     0.25                (Banana, Bread)
18     0.25                  (Apple, Milk)
19     0.25                 (Apple, Flour)
20     0.20                 (Apple, Bread)
21     0.20          (Apple, Bread, Flour)
22     0.20                 (Bread, Flour)
```



```
FP-Growth Rules:
Rule 1:  Eggs => Milk; confidence: 0.714
Rule 2:  Eggs => Banana; confidence: 0.714
Rule 3:  Apple => Milk; confidence: 0.714
Rule 4:  Apple => Flour; confidence: 0.714
Rule 5:  Flour => Apple; confidence: 0.714
Rule 6:  Lamb Meat => Banana; confidence: 0.833
Rule 7:  Apple, Bread => Flour; confidence: 1.0
Rule 8:  Apple, Flour => Bread; confidence: 0.8
Rule 9:  Bread, Flour => Apple; confidence: 1.0
Rule 10:  Lamb Meat, Banana => Bread; confidence: 0.8
Rule 11:  Lamb Meat, Bread => Banana; confidence: 1.0
Rule 12:  Banana, Bread => Lamb Meat; confidence: 0.8
FP-Growth Execution time : 0.028
Apriori is 8.0 times faster than brute force
FP-Growth is 9.0 times faster than brute force
FP-Growth is 1.0 time faster than Apriori
```

Conclusion:

- Brute force is the slowest the algorithm for associate rule mining
- All three algorithms produce same frequency items set
- All three algorithms produce same rules
- FP-Growth is the fastest algorithm for association rule mining.

THANKYOU……