**NAME OF EXPERIMENT**: Implementation FP-growth algorithm for finding frequent item set using Python.

**INTRODUCTION**: The FP-Growth algorithm is a data mining algorithm used for discovering frequent item sets in a transactional database. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree. The tree structure will maintain the association between the item sets.

Mining frequent patterns using FP-tree:

1. Start from each frequent length-1 pattern (suffix pattern).
2. Construct its conditional pattern base.
3. Construct its conditional FP-tree.
4. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

FP Tree:

An FP-tree, or frequent pattern tree, is a data structure used by the FP-Growth algorithm for efficient mining of frequent patterns in a transactional database. The FP-tree is constructed by recursively inserting each transaction in the database into the tree, creating a path from the root of the tree to a leaf node for each transaction. Each node in the tree represents an item in the transactional database, and the count of each item is stored in the corresponding node. Additionally, each node may also have a link to the next node in the tree with the same item, forming a linked list of nodes for each item.

**MANUAL SOLUTION:**

|  |  |
| --- | --- |
| Transaction ID | **Items** |
| 1 | E,K,M,N,O,Y |
| 2 | D,E,K,N,O,Y |
| 3 | A,E,K,M |
| 4 | C,K,M,U,Y |
| 5 | C,E,I,K,O,O |

Min support = 3

Step1: Find out frequency of each data set (1-itemset)

|  |  |
| --- | --- |
| Item | frequency |
| A | 1 |
| C | 2 |
| D | 1 |
| E | 4 |
| I | 1 |
| K | 5 |
| M | 3 |
| N | 2 |
| O | 3 |
| U | 1 |
| Y | 3 |

In above table item {K, E, M, O, Y} crosses the min support= 3

Now, Arrange items into descending order: {K: 5, E: 4, M: 3, O: 3 ,Y: 3}

Step2: For each transaction, the respective Ordered-itemset is built.

|  |  |  |
| --- | --- | --- |
| Transaction ID | Item sets | Ordered-item sets |
| 1 | E,K,M,N,O,Y | K,E,M,O,Y |
| 2 | D,E,K,N,O,Y | K,E,O,Y |
| 3 | A,E,K,M | K,E,M |
| 4 | C,K,M,U,Y | K,M,Y |
| 5 | C,E,I,K,O,O | K,E,O |

Step3: Building FP-tree by using Ordered-item set.

Inserting the set {K,E,M,O,Y}

NULL

K: 1 2 3 4 5

E: 1 2 3 4

M: 1

M: 1 2

O: 1 2

O: 1

Y: 1

Y: 1

Y: 1

Step4: Conditional Pattern

|  |  |  |
| --- | --- | --- |
| Item | Conditional Pattern | Conditional FP-tree |
| Y | {K,E,M,O: 1},{K,E,O: 1},{K,M: 1} | K=3 |
| O | {K,E,M: 1},{K,E: 2} | K,E=3 |
| M | {K,E: 2}, {K:1} | K=3 |
| E | {K: 4} | K=4 |
| K | --- | --- |

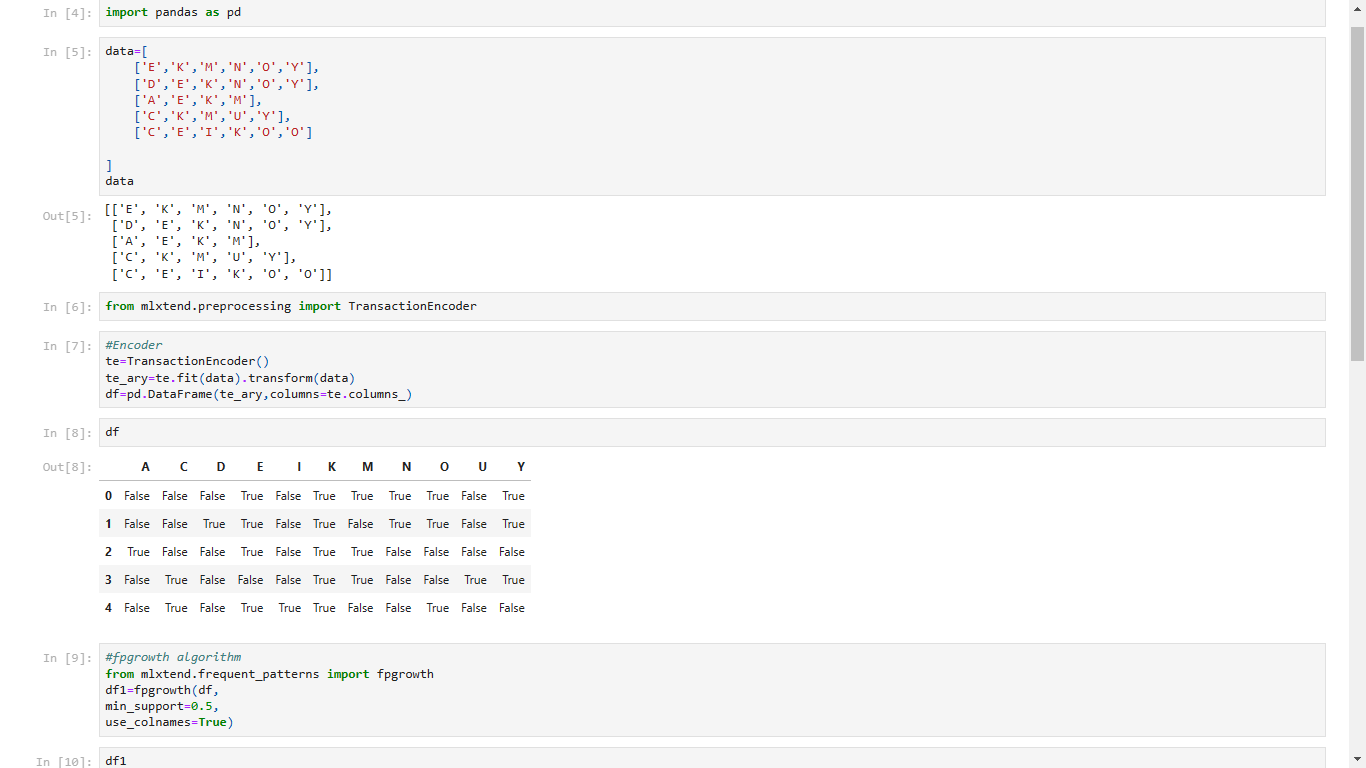
ADVANTAGES OF FP-GROWTH ALGORITHM: -

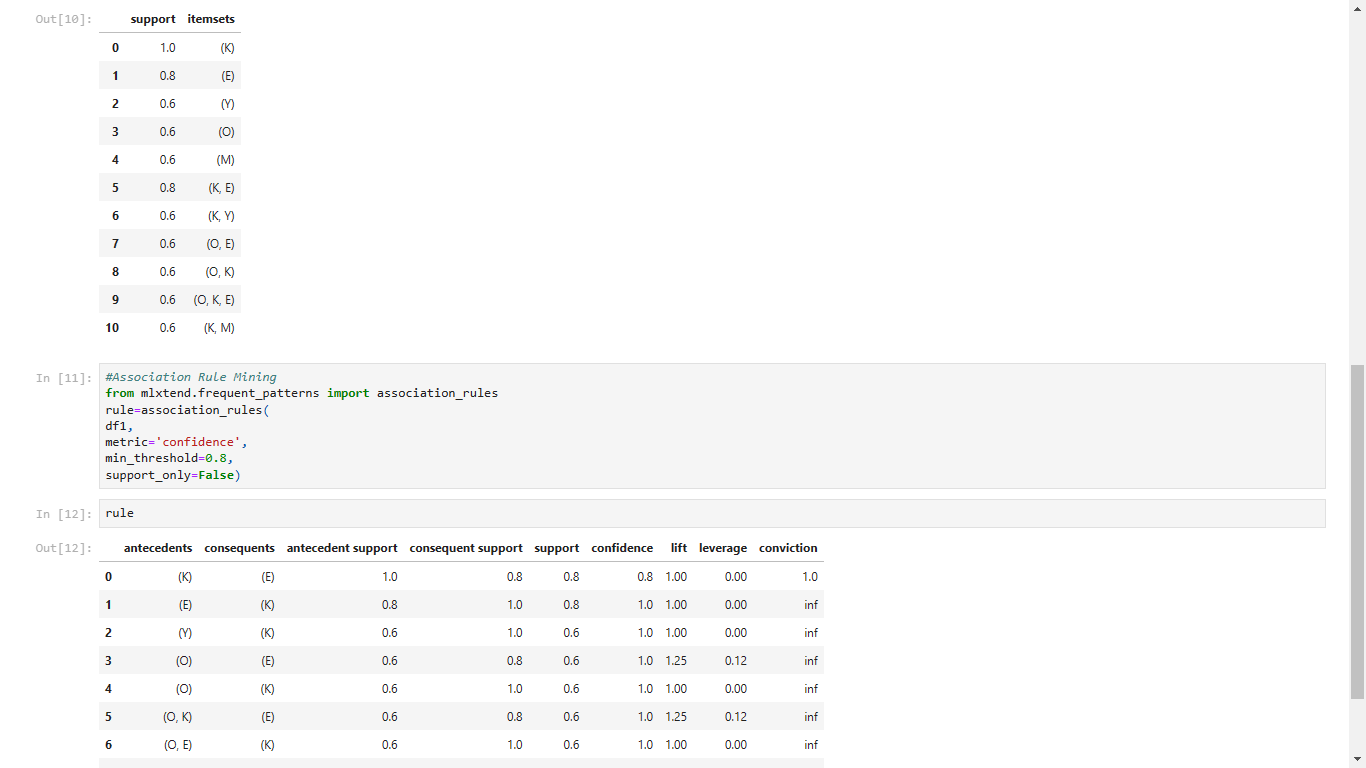
1. No candidate generation, no candidate test.
2. Use compact data structure called FP- Tree
3. It is faster than Apriori algorithm.

DISADVANTAGES OF FP-GROWTH ALGORITHM: -

1. FP-tree is difficult to build than Apriori.
2. It may be expensive.
3. FP-tree may not fit in the memory

**PROGRAM IMPLEMENTATION IN PYTHON**

Source Code: 



CONCLUSION: -

Hence, FP-growth algorithm using Python successfully implemented.