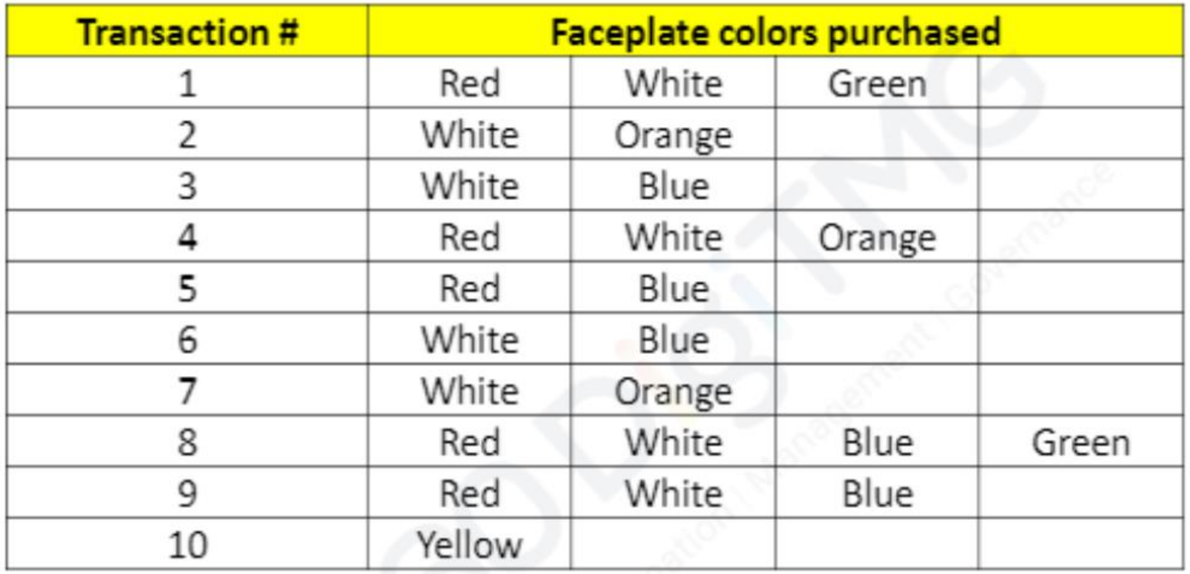
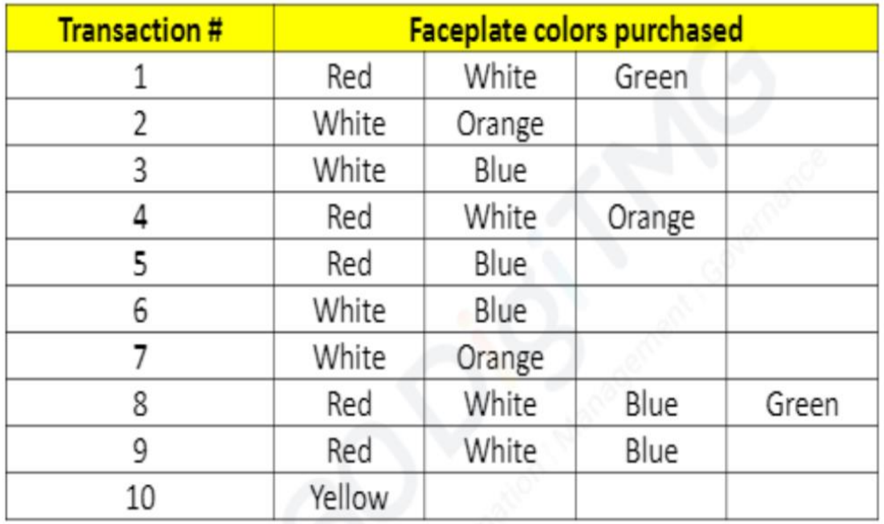


Transaction #	Faceplate colors purchased			
1	Red	White	Green	
2	White	Orange		
3	White	Blue		
4	Red	White	Orange	
5	Red	Blue		
6	White	Blue		
7	White	Orange		
8	Red	White	Blue	Green
9	Red	White	Blue	
10	Yellow			

- What is the ***support*** for “if White then Blue”?
  1. 4
  2. 40%
  3. 2
  4. 90%

- What is the ***support*** for “if Blue then White”?
  1. 4
  2. 40%
  3. 2
  4. 90%



Transaction #	Faceplate colors purchased			
1	Red	White	Green	
2	White	Orange		
3	White	Blue		
4	Red	White	Orange	
5	Red	Blue		
6	White	Blue		
7	White	Orange		
8	Red	White	Blue	Green
9	Red	White	Blue	
10	Yellow			

- What is the ***confidence*** for "if White then Blue"?
  1. 4/5
  2. 5/8
  3. 5/4
  4. 4/8

- What is the ***confidence*** for "if Blue then White"?
  1. 4/5
  2. 5/8
  3. 5/4
  4. 4/8

For k products:

1. Set minimum support criteria
2. Generate list of one-item sets that meet the support criterion
3. Use list of one-item sets to generate list of two-item sets that meet support criterion
4. Use list of two-item sets to generate list of three-item sets that meet support criterion
5. Continue up through k-item sets

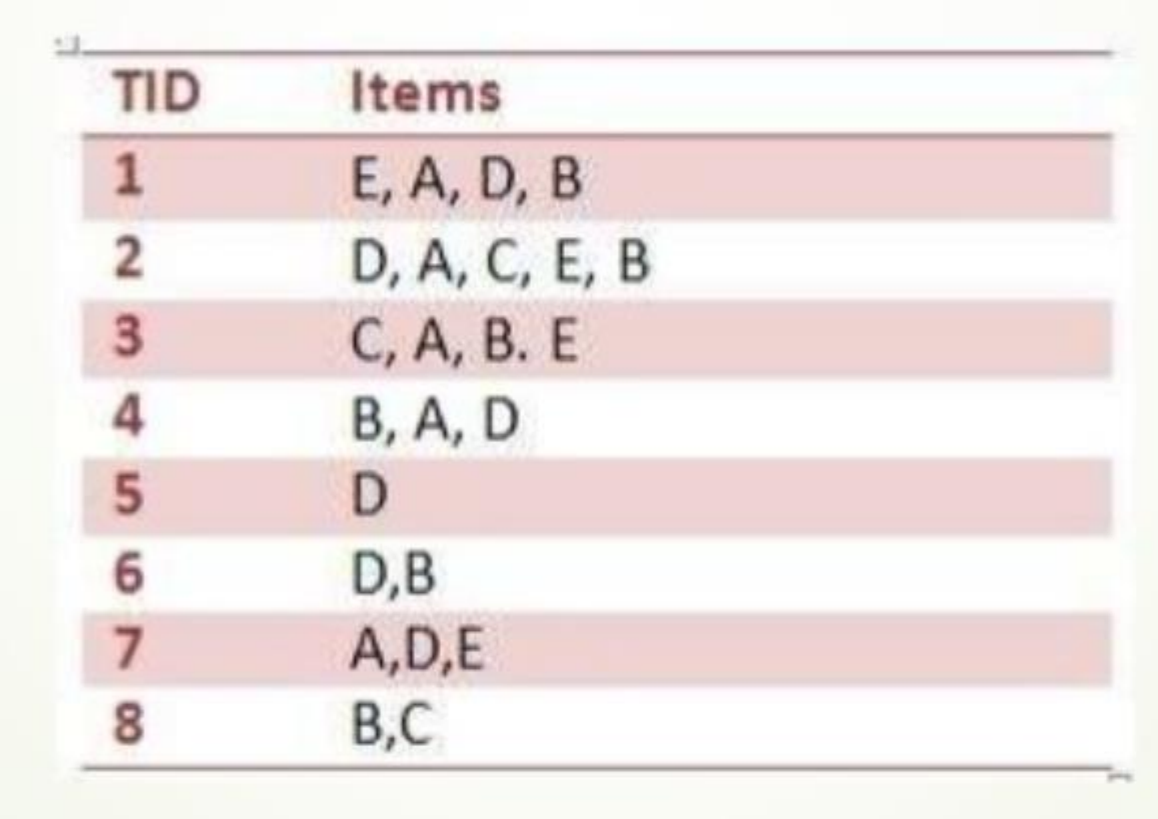
Support Min Criterion = 2

Transaction #	Faceplate colors purchased			
1	Red	White	Green	
2	White	Orange		
3	White	Blue		
4	Red	White	Orange	
5	Red	Blue		
6	White	Blue		
7	White	Orange		
8	Red	White	Blue	Green
9	Red	White	Blue	
10	Yellow			

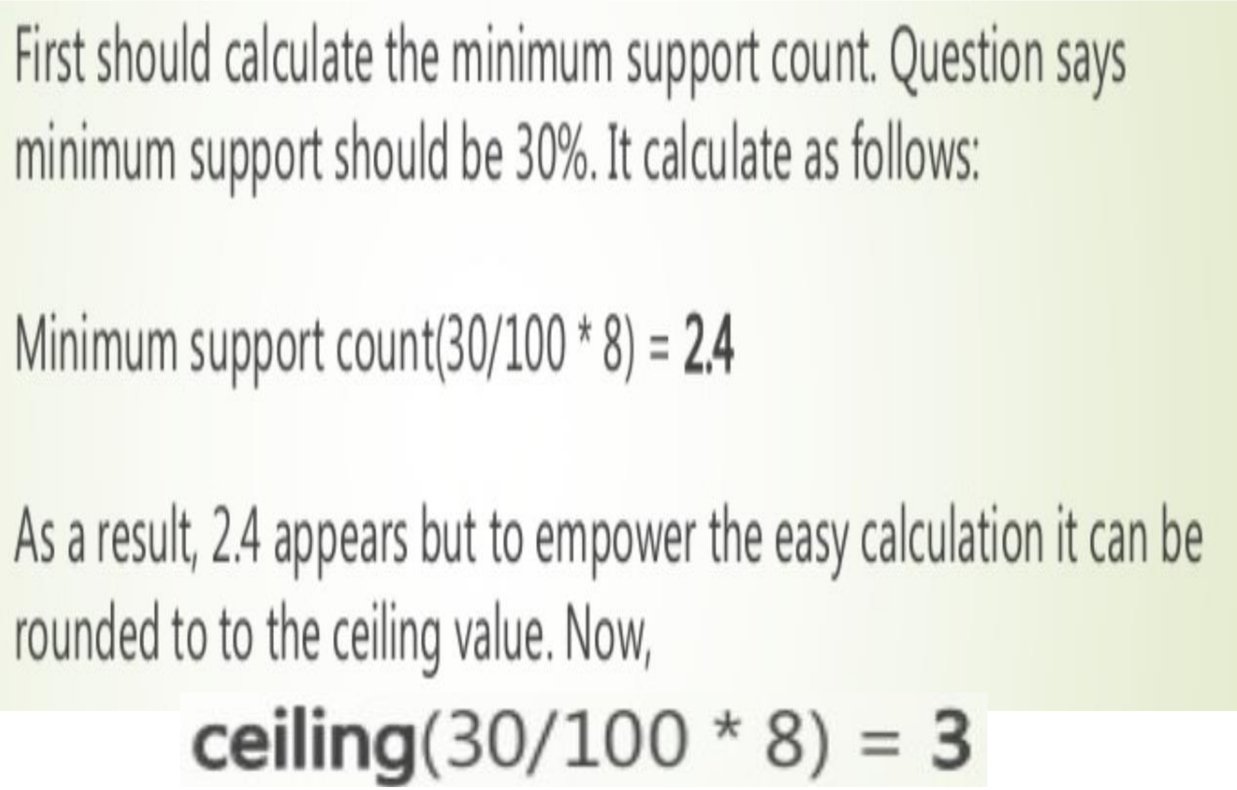
Item set	Support (Count)
{Red}	5
{White}	8
{Blue}	5
{Orange}	3
{Green}	2
{Red, White}	4
{Red, Blue}	3
{Red, Green}	2
{White, Blue}	4
{White, Orange}	3
{White, Green}	2
{Red, White, Blue}	2
{Red, White, Green}	2

Create rules from frequent item sets only

STEP-1: CALCULATE MINIMUM SUPPORT COUNT/FREQUENCY



TID	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B. E
4	B, A, D
5	D
6	D,B
7	A,D,E
8	B,C



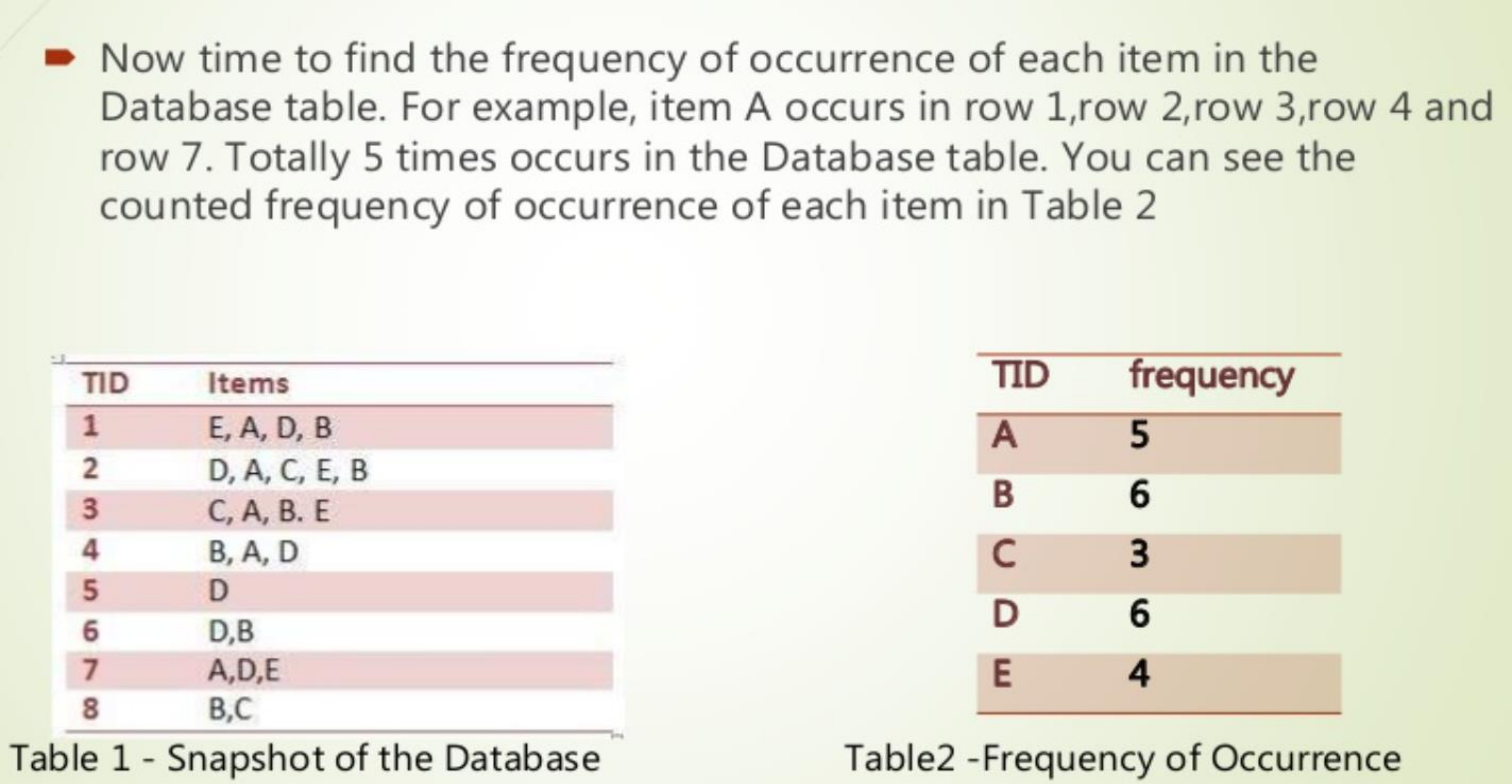
First should calculate the minimum support count. Question says minimum support should be 30%. It calculate as follows:

Minimum support count(30/100 * 8) = **2.4**

As a result, 2.4 appears but to empower the easy calculation it can be rounded to to the ceiling value. Now,

ceiling(30/100 * 8) = **3**

STEP-2: FIND THE FREQUENCY OF OCCURENCE (REMOVE THE ONES WHICH IS LESS THAN MIN SUPPORT COUNT)



- Now time to find the frequency of occurrence of each item in the Database table. For example, item A occurs in row 1,row 2,row 3,row 4 and row 7. Totally 5 times occurs in the Database table. You can see the counted frequency of occurrence of each item in Table 2

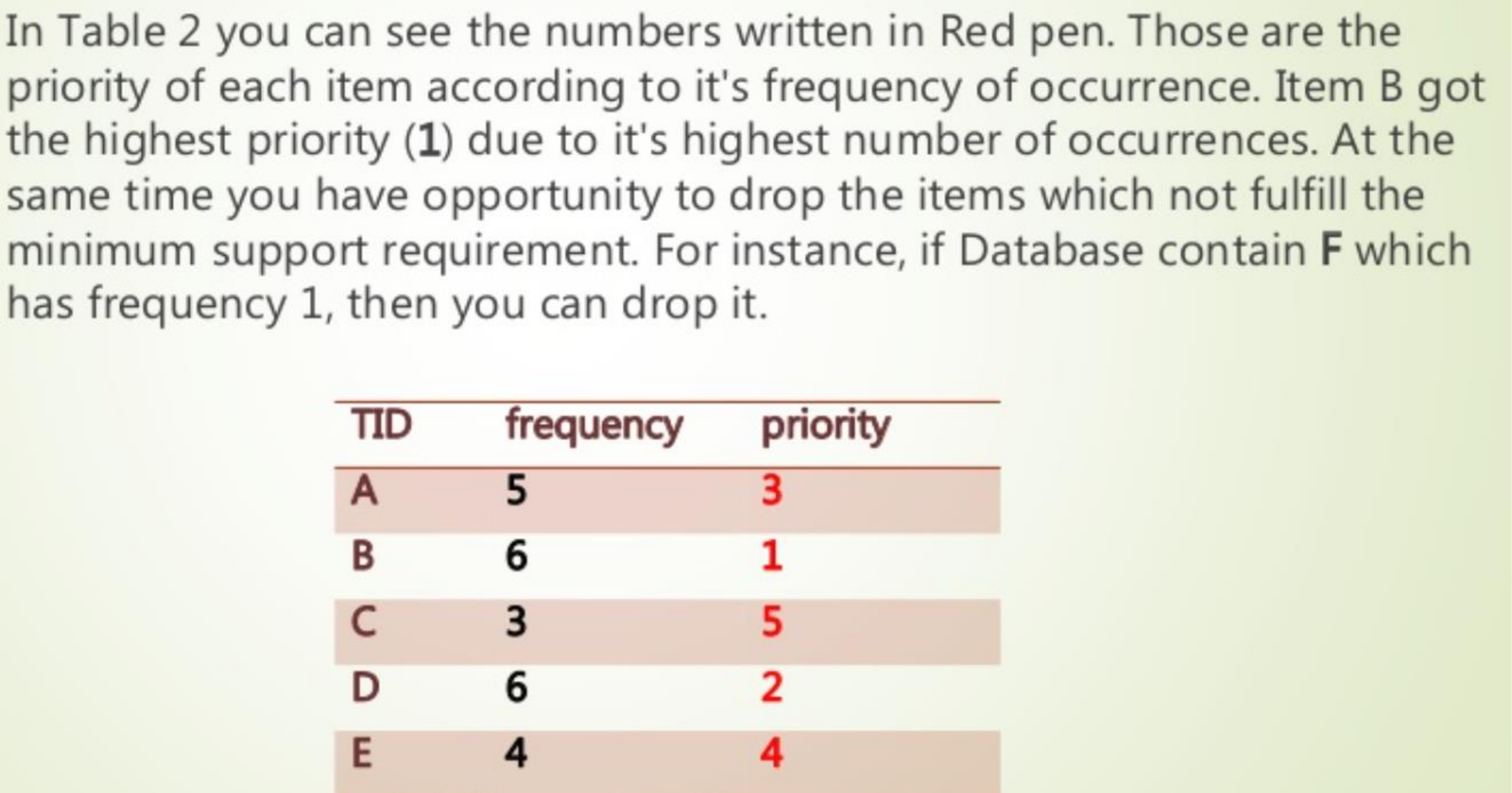
TID	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B. E
4	B, A, D
5	D
6	D,B
7	A,D,E
8	B,C

Table 1 - Snapshot of the Database

TID	frequency
A	5
B	6
C	3
D	6
E	4

Table2 -Frequency of Occurrence

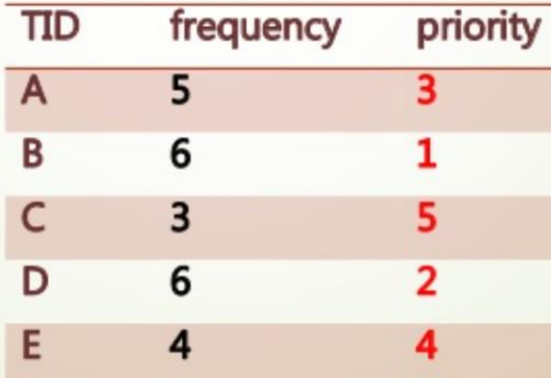
STEP-3: PRIORITIZE THE ITEMS (ORDER...HIGH TO LOW)



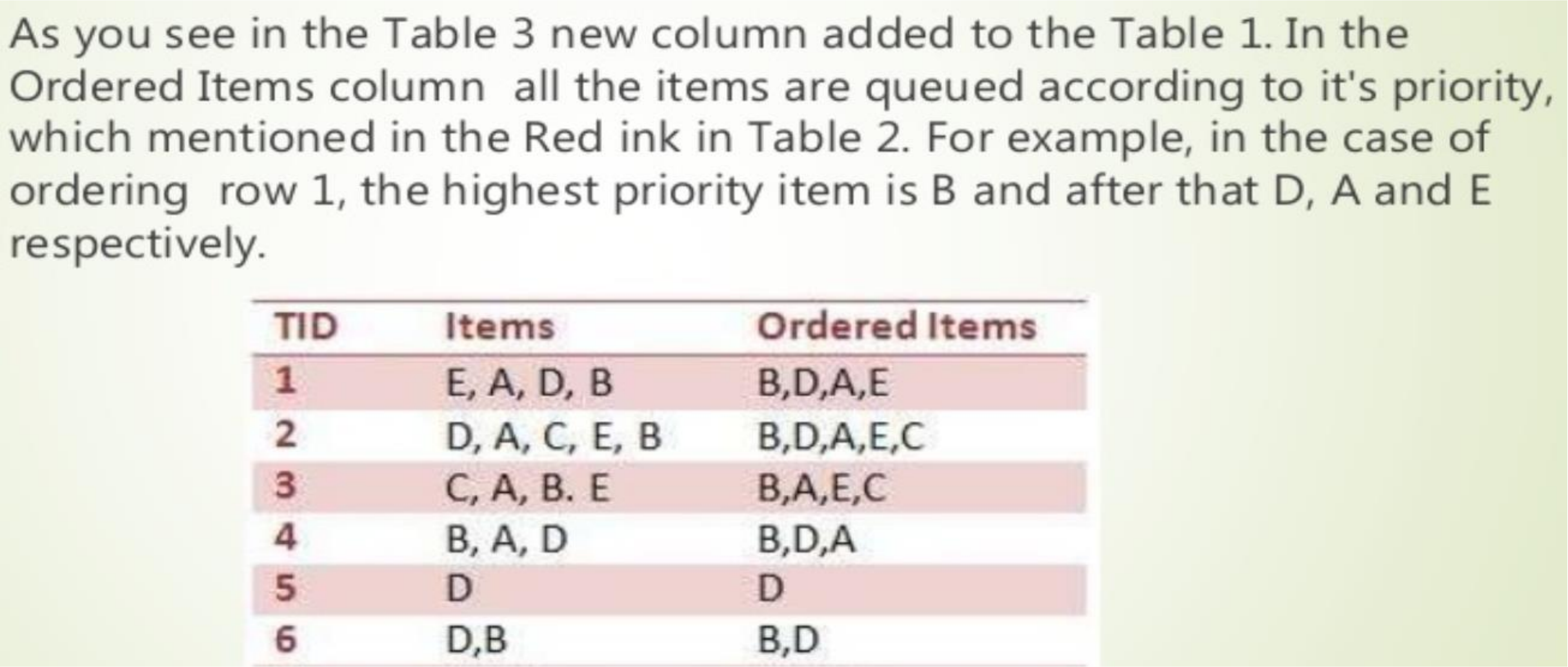
In Table 2 you can see the numbers written in Red pen. Those are the priority of each item according to it's frequency of occurrence. Item B got the highest priority (**1**) due to it's highest number of occurrences. At the same time you have opportunity to drop the items which not fulfill the minimum support requirement. For instance, if Database contain **F** which has frequency 1, then you can drop it.

TID	frequency	priority
A	5	3
B	6	1
C	3	5
D	6	2
E	4	4

STEP-4: ORDER THE ITEMS ACCORDING TO PRIORITY



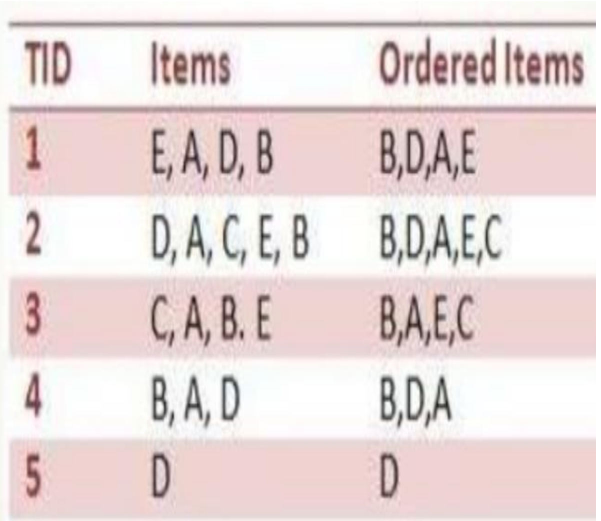
TID	frequency	priority
A	5	3
B	6	1
C	3	5
D	6	2
E	4	4



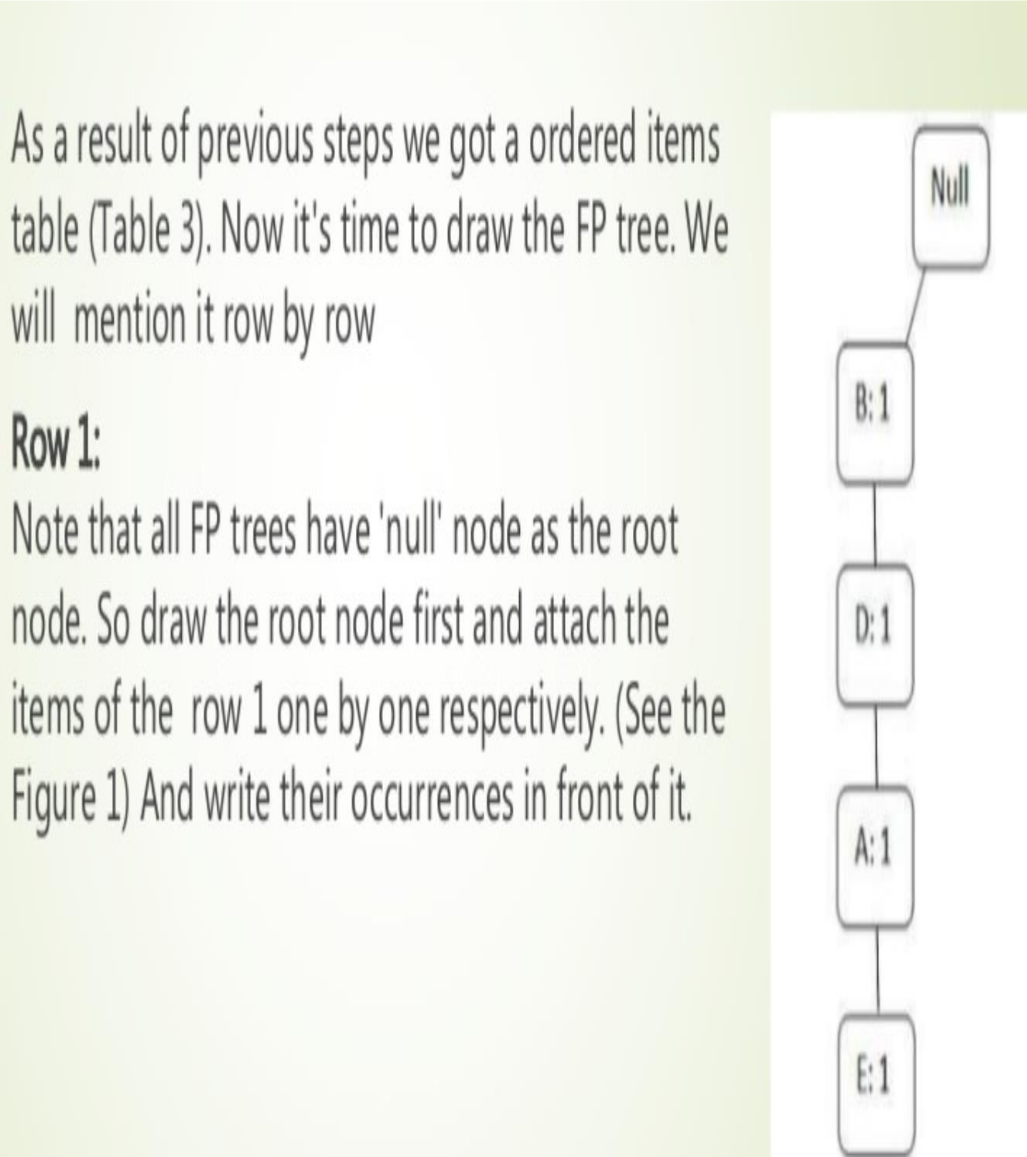
As you see in the Table 3 new column added to the Table 1. In the Ordered Items column all the items are queued according to it's priority, which mentioned in the Red ink in Table 2. For example, in the case of ordering row 1, the highest priority item is B and after that D, A and E respectively.

TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D

STEP-5: ORDER THE ITEMS ACCORDING TO PRIORITY



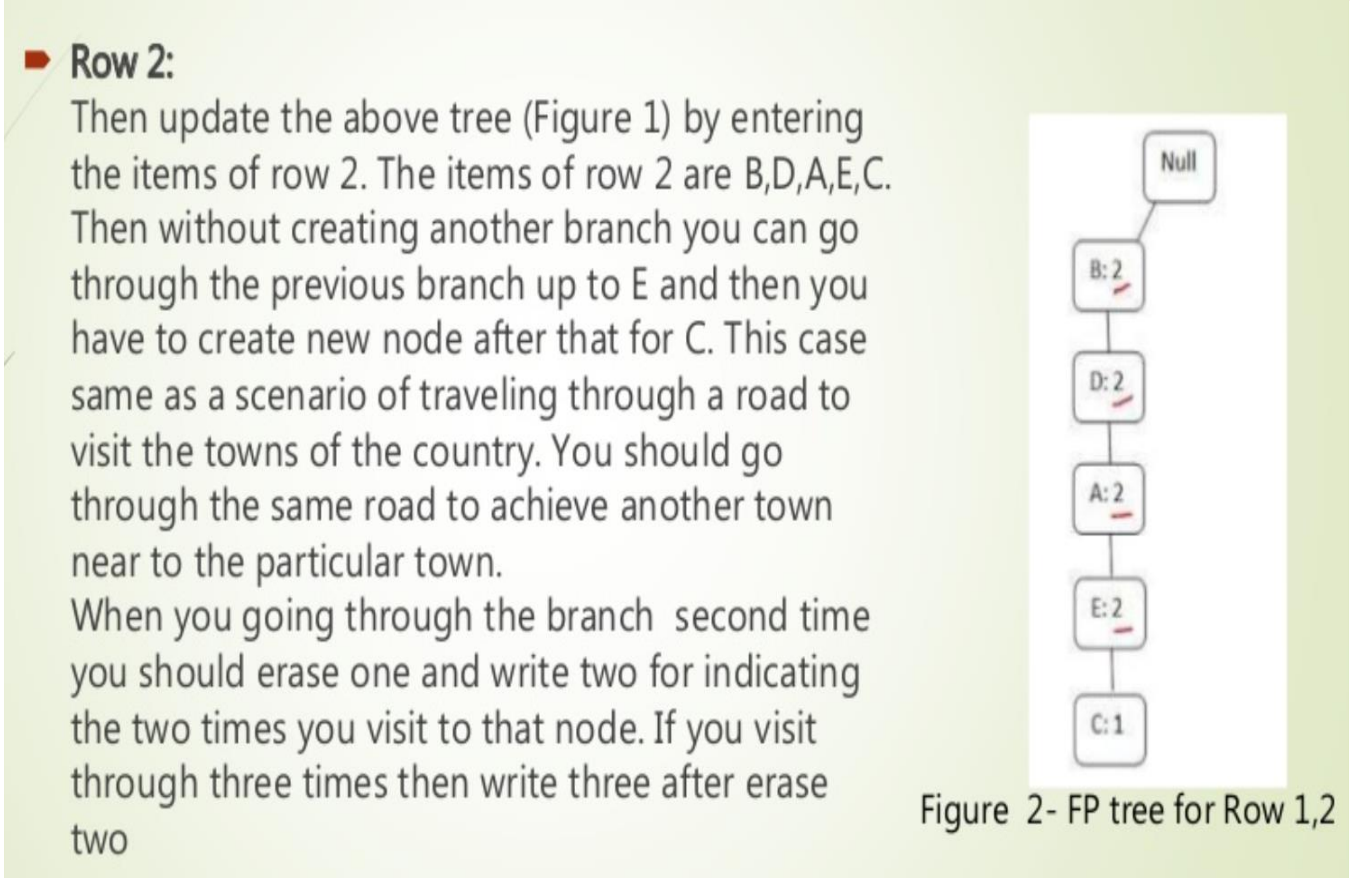
TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D



As a result of previous steps we got a ordered items table (Table 3). Now it's time to draw the FP tree. We will mention it row by row

Row 1:

Note that all FP trees have 'null' node as the root node. So draw the root node first and attach the items of the row 1 one by one respectively. (See the Figure 1) And write their occurrences in front of it.

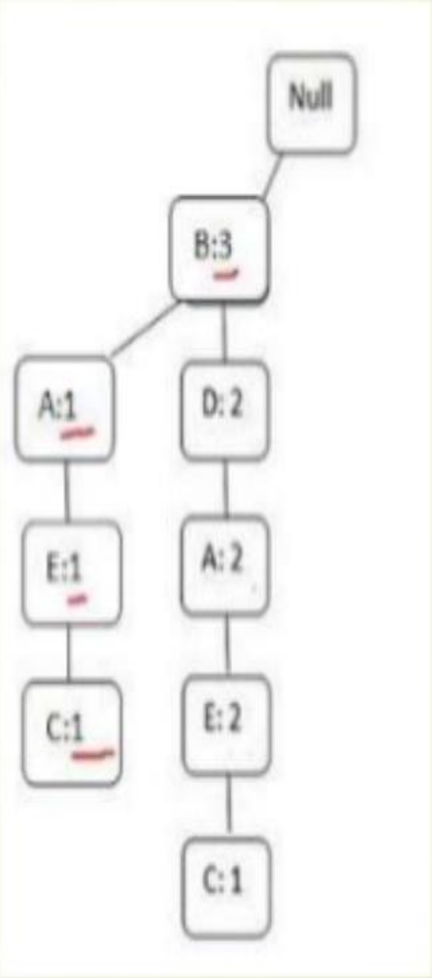


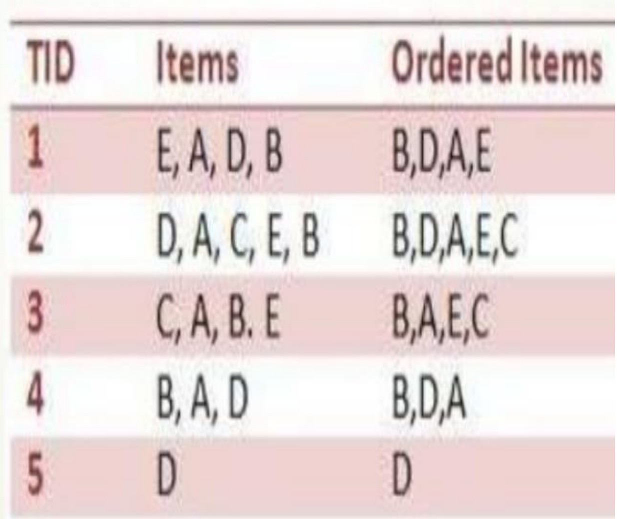
- **Row 2:**
  Then update the above tree (Figure 1) by entering the items of row 2. The items of row 2 are B,D,A,E,C. Then without creating another branch you can go through the previous branch up to E and then you have to create new node after that for C. This case same as a scenario of traveling through a road to visit the towns of the country. You should go through the same road to achieve another town near to the particular town.
  When you going through the branch second time you should erase one and write two for indicating the two times you visit to that node. If you visit through three times then write three after erase two

Figure 2- FP tree for Row 1,2

Row 3:

In row 3 you have to visit B,A,E and C respectively. So you may think you can follow the same branch again by replacing the values of B,A,E and C . But you can't do that you have opportunity to come through the B. But can't connect B to existing A overtaking D. As a result you should draw another A and connect it to B and then connect new E to that A and new C to new E.

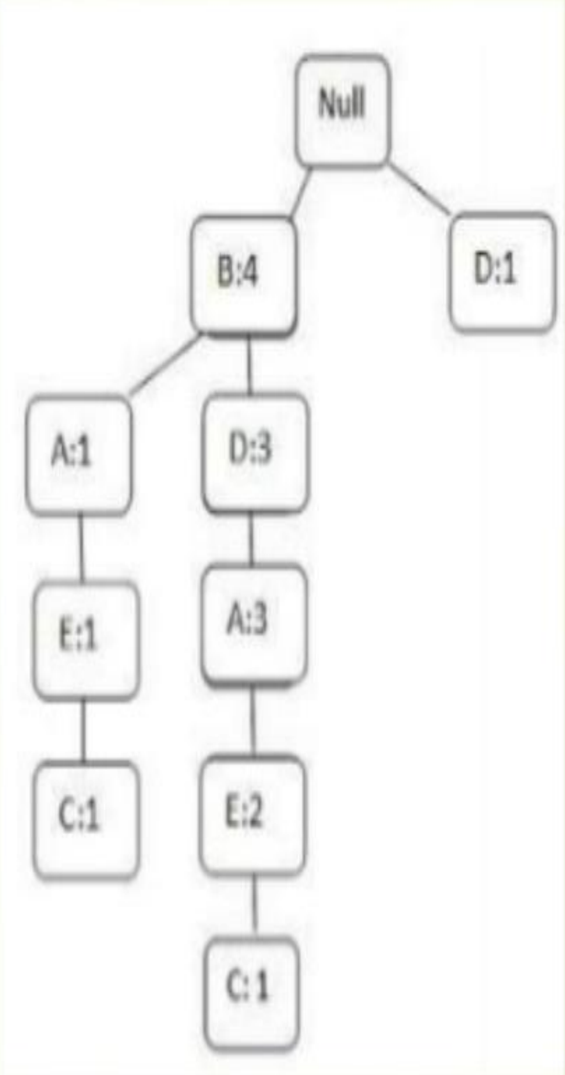


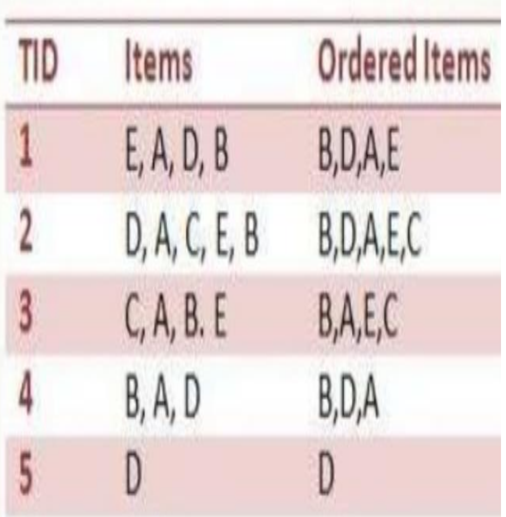


TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D

- **Row 4:**
  Then row 4 contain B,D,A. Now we can just rename the frequency of occurrences in the existing branch. As B:4,D,A:3.

- **Row 5:**
  n fifth raw have only item D. Now we have opportunity draw new branch from 'null' node. See Figure 4.





TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D

- **Row 6:**
  B and D appears in row 6. So just change the B:4 to B:5 and D:3 to D:4.

- **Row 7:**
  Attach two new nodes A and E to the D node which hanging on the null node. Then mark D,A,E as D:2,A:1 and E:1.

- **Row 8 :(Ohh.. last row)**
  Attach new node C to B. Change the traverse times.(B:6,C:1

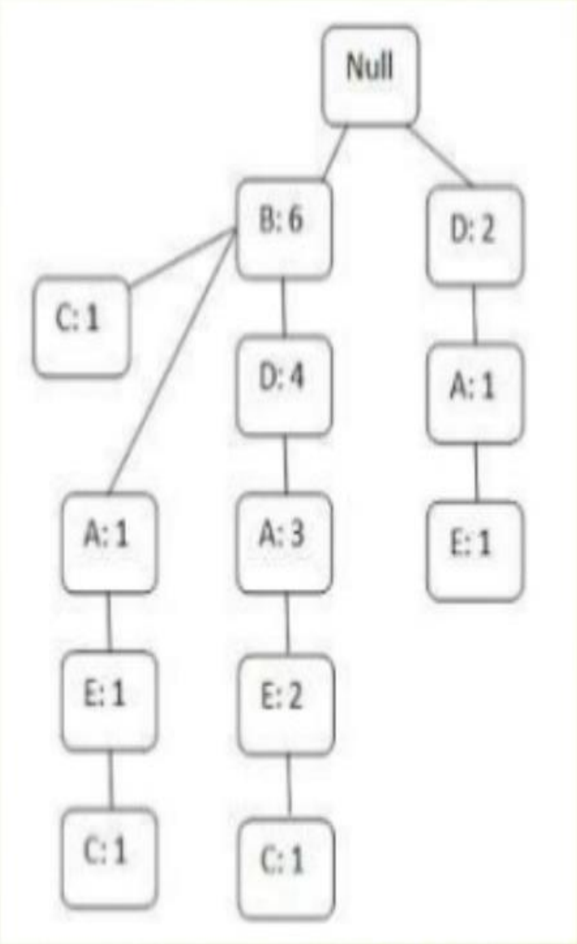
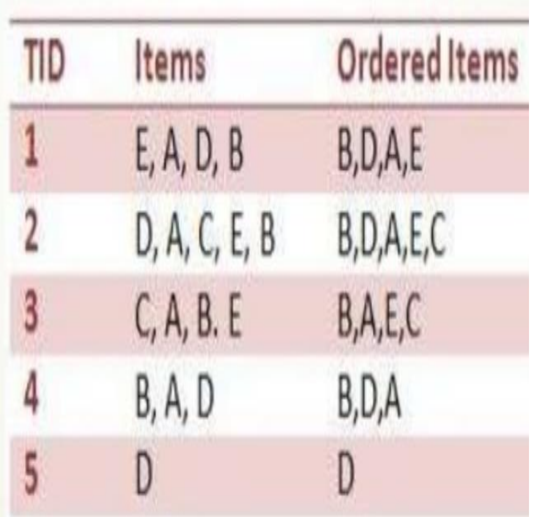
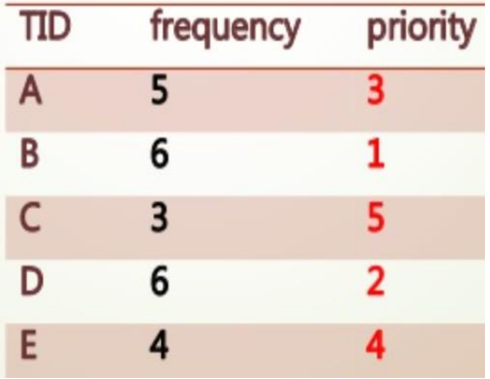


Figure 5 - Final FP tree



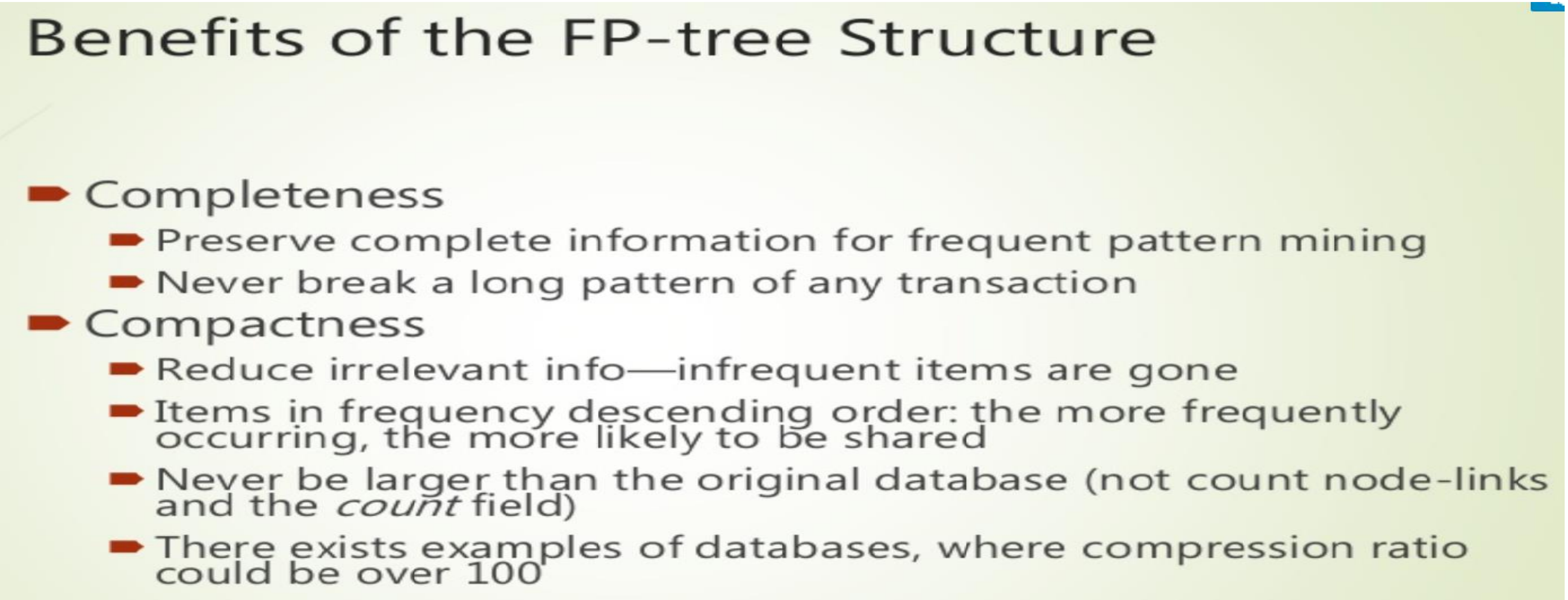
TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D



TID	frequency	priority
A	5	3
B	6	1
C	3	5
D	6	2
E	4	4

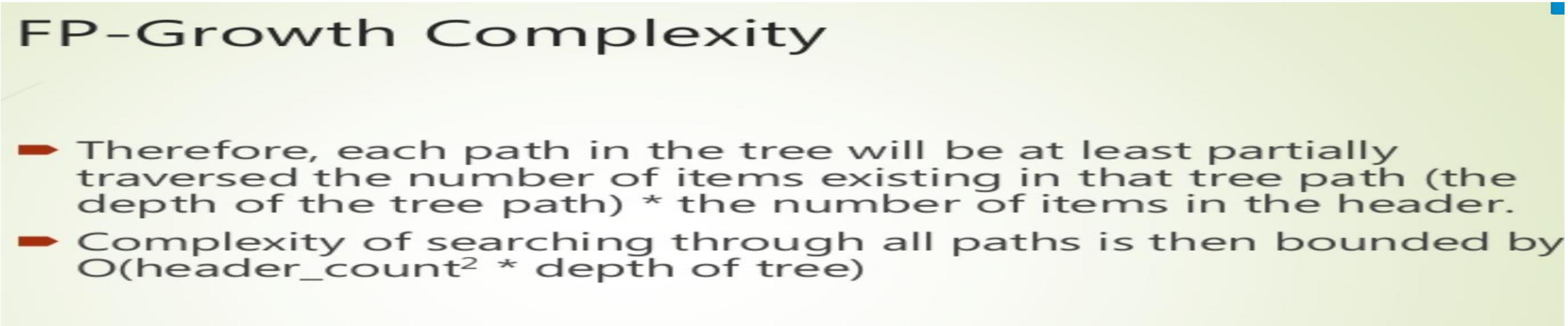
How we know is this correct?

Now count the frequency of occurrence of each item of the FP tree and compare it with Table 2. If both counts equal, then it is positive point to indicate your tree is correct.



Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - There exists examples of databases, where compression ratio could be over 100



FP-Growth Complexity

- Therefore, each path in the tree will be at least partially traversed the number of items existing in that tree path (the depth of the tree path) * the number of items in the header.
- Complexity of searching through all paths is then bounded by O(header_count^2 * depth of tree)