The **Apriori Property** is a fundamental principle used in association rule mining and the Apriori algorithm. It states:

> **"If an itemset is frequent, then all of its subsets must also be frequent."**

Explanation:

- If an itemset does not meet the minimum support threshold, none of its supersets can be frequent.
- This property helps reduce the search space by eliminating candidate itemsets that have infrequent subsets.

Example:

1. Suppose `{A, B, C}` is a frequent itemset.
2. Then, its subsets `{A, B}`, `{A, C}`, `{B, C}`, `{A}`, `{B}`, and `{C}` must also be frequent.
3. If `{A, B}` is **not** frequent, then `{A, B, C}` cannot be frequent.

This property is crucial for optimizing the Apriori algorithm, as it avoids generating and checking unnecessary candidate itemsets.

To generate the **3-itemset** from the **2-itemset**, follow these systematic steps:

Step-by-Step Process

1. **Start with the 2-itemsets**
   - You have pairs of items and their corresponding transaction lists.
2. **Find common transactions**
   - For each possible combination of two 2-itemsets that share a common item, take their intersection.
3. **Form new 3-itemsets**
   - If two 2-itemsets `{X, Y}` and `{X, Z}` share the item `X`, then they can be combined into `{X, Y, Z}`.
   - The transactions in the new 3-itemset will be the intersection of the transactions in the original 2-itemsets.

Example Walkthrough

Given 2-itemsets and Transactions

Itemset	Transactions
{11, 12}	{T1, T4, T8, T9}
{11, 13}	{T5, T7, T8, T9}
{11, 15}	{T1, T8}
{12, 13}	{T3, T6, T8, T9}
{12, 14}	{T2, T4}
{12, 15}	{T1, T8}

Step 1: Find Common Transactions

Find two pairs that share the same first item and intersect their transaction sets:

Combination	Intersection of Transactions	New 3-itemset
{11,12} ∩ {11,13}	{T8, T9}	{11,12,13}
{11,12} ∩ {11,15}	{T1, T8}	{11,12,15}

{11,13} ∩ {11,15}	{T8}	{11,13,15}
{12,13} ∩ {12,15}	{T8}	{12,13,15}
{12,13} ∩ {12,14}	**(empty)**	**N/A**
{12,14} ∩ {12,15}	**(empty)**	**N/A**

Final 3-itemsets

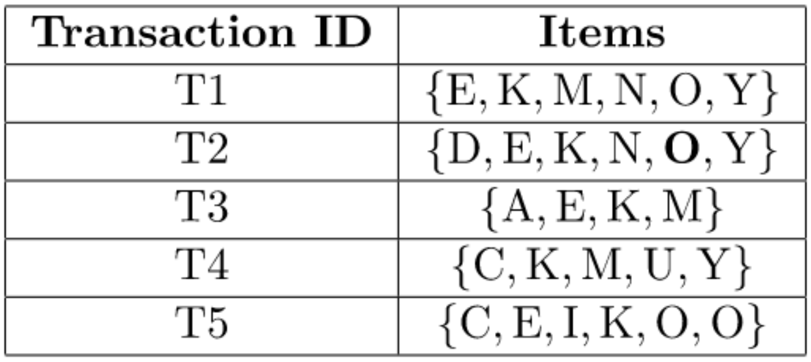
3-itemset	Transactions
{11,12,13}	{T8, T9}
{11,12,15}	{T1, T8}
{11,13,15}	{T8}
{12,13,15}	{T8}

How to Do This Easily?

1. **Automate with Python/Pandas:**
   - Store 2-itemsets and their transactions in a dictionary.
   - Use set intersection to find transactions common between pairs of 2-itemsets.
   - Generate the 3-itemsets dynamically.

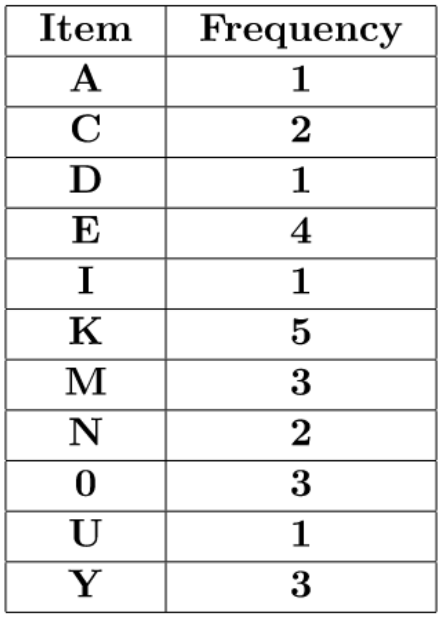
Example-1

Consider the following data:-



Transaction ID	Items
T1	{E, K, M, N, O, Y}
T2	{D, E, K, N, **O**, Y}
T3	{A, E, K, M}
T4	{C, K, M, U, Y}
T5	{C, E, I, K, O, O}

The above-given data is a hypothetical dataset of transactions with each letter representing an item. The frequency of each individual item is computed:-



Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
0	3
U	1
Y	3

Let the **minimum support be 3**. A **Frequent Pattern set** is built which will contain all the elements whose frequency is greater than or equal to the minimum support. These elements are stored in descending order of their respective frequencies. After insertion of the relevant items, the set L looks like this:-

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

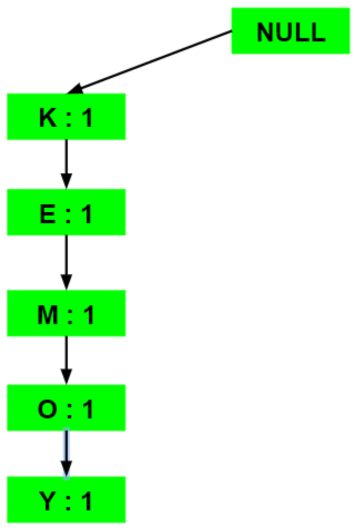
Now, for each transaction, the respective **Ordered-Item set** is built. It is done by iterating the Frequent Pattern set and checking if the current item is contained in the transaction in question. If the current item is contained, the item is inserted in the Ordered-Item set for the current transaction. The following table is built for all the transactions:

Transaction ID	Items	Ordered-Item Set
T1	{E, K, M, N, O, Y}	{K, E, M, O, Y}
T2	{D, E, K, N, O, Y}	{K, E, O, Y}
T3	{ A, E, K, M}	{K, E, M}
T4	{C, K, M, U, Y}	{K, M, Y}
T5	{C, E, I, K, O, O}	{K, E, O}

Now, all the Ordered-Item sets are inserted into a Trie Data Structure.

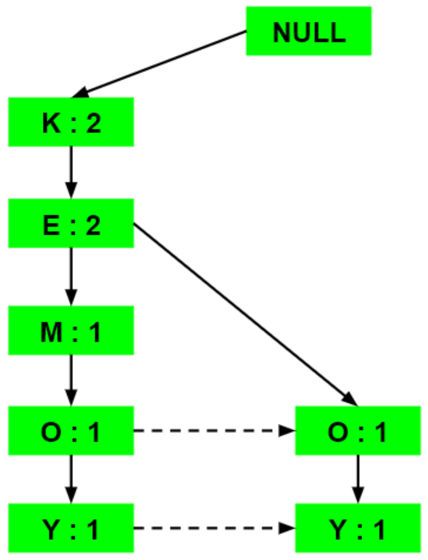
a) **Inserting the set {K, E, M, O, Y}:**

Here, all the items are simply linked one after the other in the order of occurrence in the set and initialize the support count for each item as 1.



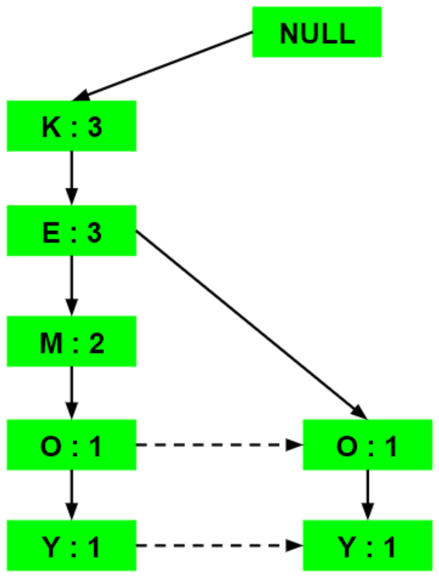
b) **Inserting the set {K, E, O, Y}:**

Till the insertion of the elements K and E, simply the support count is increased by 1. On inserting O we can see that there is no direct link between E and O, therefore a new node for the item O is initialized with the support count as 1 and item E is linked to this new node. On inserting Y, we first initialize a new node for the item Y with support count as 1 and link the new node of O with the new node of Y.



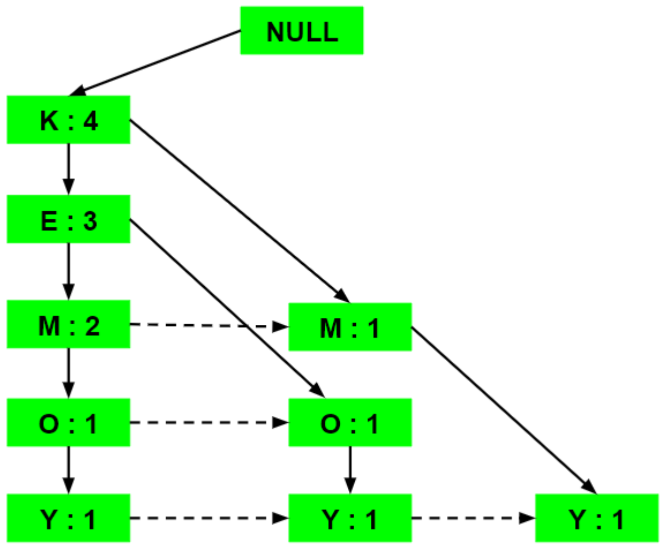
c) **Inserting the set {K, E, M}:**

Here simply the support count of each element is increased by 1.



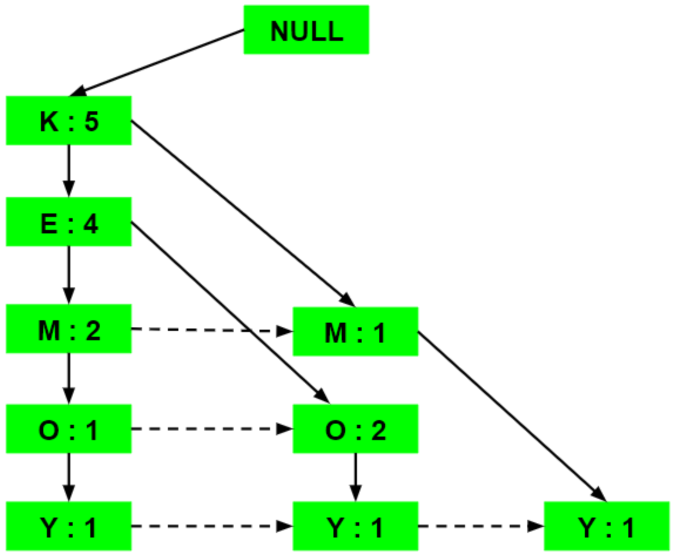
d) **Inserting the set {K, M, Y}:**

Similar to step b), first the support count of K is increased, then new nodes for M and Y are initialized and linked accordingly.



e) **Inserting the set {K, E, O}:**

Here simply the support counts of the respective elements are increased. Note that the support count of the new node of item O is increased.



Now, for each item, the **Conditional Pattern Base** is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree. Note that the items in the below table are arranged in the ascending order of their frequencies.

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

Now for each item, the **Conditional Frequent Pattern Tree is built.** It is done by taking the set of elements that is common in all the paths in the Conditional Pattern Base of that item and calculating its support count by summing the support counts of all the paths in the Conditional Pattern Base.

Items	Conditional Pattern Base	Conditional Frequent Pattern 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		

From the Conditional Frequent Pattern tree, the **Frequent Pattern rules** are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding to the item as given in the below table.

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 4>}
K	

For each row, two types of association rules can be inferred for example for the first row which contains the element, the rules K -> Y and Y -> K can be inferred. To determine the valid rule, the confidence of both the rules is calculated and the one with confidence greater than or equal to the minimum confidence value is retained.

Example-2

Support threshold=50%

Table 1

Transaction	**List of items**
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution:

Support threshold=50% => 0.5*6= 3 => min_sup=3

1. Count of each item

Table 2

Item	**Count**
I1	4
I2	5
I3	4
I4	4
I5	2

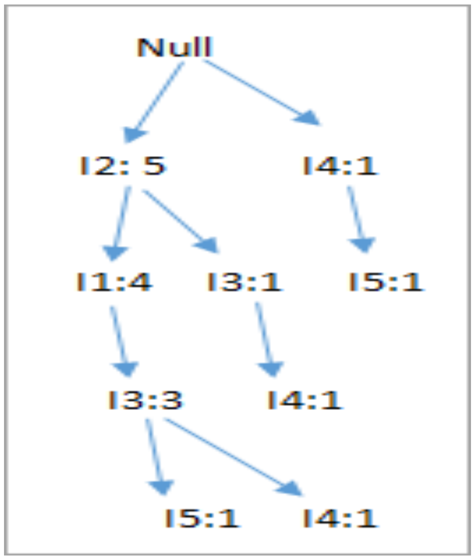
2. Sort the itemset in descending order.

Table 3

Item	**Count**
I2	5
I1	4
I3	4
I4	4

3. Build FP Tree

1. Considering the root node null.
2. The first scan of Transaction T1: I1, I2, I3 contains three items {I1:1}, {I2:1}, {I3:1}, where I2 is linked as a child to root, I1 is linked to I2 and I3 is linked to I1.
3. T2: I2, I3, I4 contains I2, I3, and I4, where I2 is linked to root, I3 is linked to I2 and I4 is linked to I3. But this branch would share I2 node as common as it is already used in T1.
4. Increment the count of I2 by 1 and I3 is linked as a child to I2, I4 is linked as a child to I3. The count is {I2:2}, {I3:1}, {I4:1}.
5. T3: I4, I5. Similarly, a new branch with I5 is linked to I4 as a child is created.
6. T4: I1, I2, I4. The sequence will be I2, I1, and I4. I2 is already linked to the root node, hence it will be incremented by 1. Similarly I1 will be incremented by 1 as it is already linked with I2 in T1, thus {I2:3}, {I1:2}, {I4:1}.
7. T5:I1, I2, I3, I5. The sequence will be I2, I1, I3, and I5. Thus {I2:4}, {I1:3}, {I3:2}, {I5:1}.
8. T6: I1, I2, I3, I4. The sequence will be I2, I1, I3, and I4. Thus {I2:5}, {I1:4}, {I3:3}, {I4 1}.



4. Mining of FP-tree is summarized below:

1. The lowest node item I5 is not considered as it does not have a min support count, hence it is deleted.
2. The next lower node is I4. I4 occurs in 2 branches , {I2,I1,I3:,I41},{I2,I3,I4:1}. Therefore considering I4 as suffix the prefix paths will be {I2, I1, I3:1}, {I2, I3: 1}. This forms the conditional pattern base.
3. The conditional pattern base is considered a transaction database, an FP-tree is constructed. This will contain {I2:2, I3:2}, I1 is not considered as it does not meet the min support count.
4. This path will generate all combinations of frequent patterns : {I2,I4:2},{I3,I4:2},{I2,I3,I4:2}

5. For I3, the prefix path would be: {I2,I1:3},{I2:1}, this will generate a 2 node FP-tree : {I2:4, I1:3} and frequent patterns are generated: {I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}.
6. For I1, the prefix path would be: {I2:4} this will generate a single node FP-tree: {I2:4} and frequent patterns are generated: {I2, I1:4}.

Item	**Conditional Pattern Base**	**Conditional FP-tree**	**Frequent Patterns Generated**
I4	{I2,I1,I3:1},{I2,I3:1}	{I2:2, I3:2}	{I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}
I1	{I2:4}	{I2:4}	{I2,I1:4}

Example-3

Consider the below dataset. The minimum support given is 3.

TID	Items Bought
100	*f, a, c, d, g, i, m, p*
200	*a, b, c, f, l, m, o*
300	*b, f, h, j, o*
400	*b, c, k, s, p*
500	*a, f, c, e, l, p, m, n*

In the frequent pattern growth algorithm, first, we find the frequency of each item. The following table gives the frequency of each item in the given data.

See also Name Node Federation Checkpoint Backup and Snapshots

Item	Frequency	Item	Frequency
a	3	j	1
b	3	k	1
c	4	l	2
d	1	m	3
e	1	n	1
f	4	o	2
g	1	p	3
h	1	s	1
i	1		

A **Frequent Pattern set (L)** is built, which will contain all the elements whose frequency is greater than or equal to the minimum support.

These elements are stored in descending order of their respective frequencies.

As minimum support is 3.

After insertion of the relevant items, the set L looks like this:-

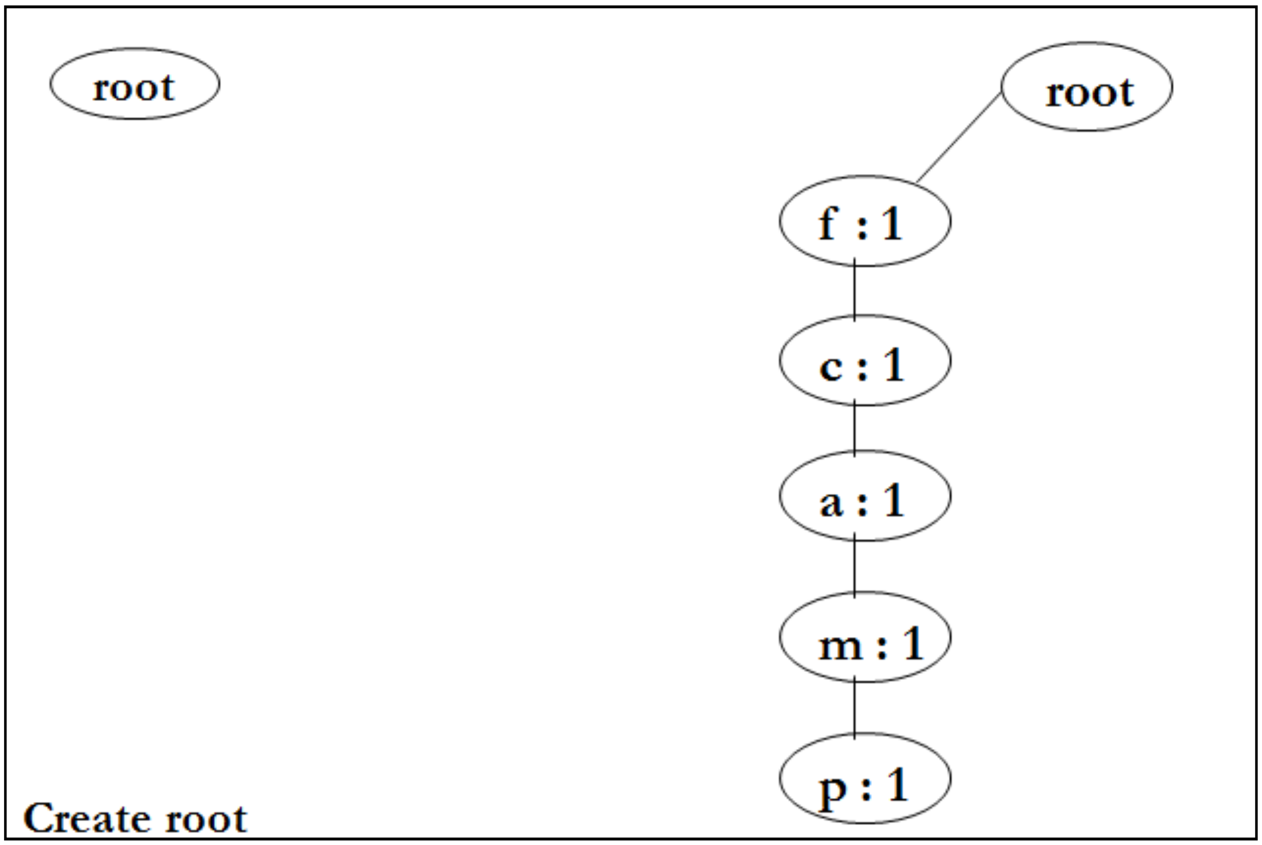
L = { (f:4), (c:4), (a:3), (b:3), (m:3), (p:3) }

Now, for each transaction, the respective **Ordered-Item set** is built.

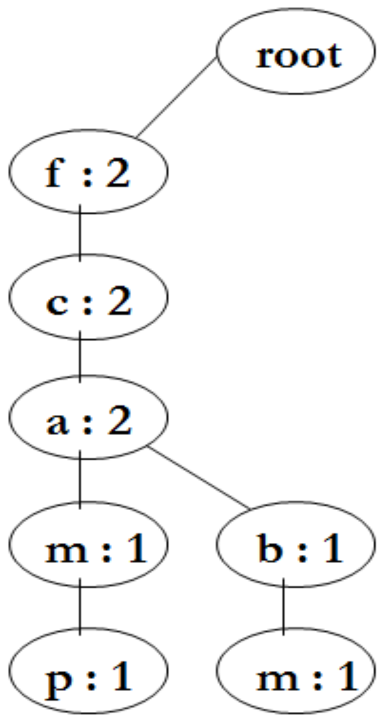
Frequent Pattern set L = { (f:4), (c:4), (a:3), (b:3), (m:3), (p:3) }

TID	**Items Bought**	**(Ordered) Frequent Items**
100	***f, a, c, d, g, i, m, p***	***f, c, a, m, p***
200	***a, b, c, f, l, m, o***	***f, c, a, b, m***
300	***b, f, h, j, o***	***f, b***
400	***b, c, k, s, p***	***c, b, p***
500	***a, f, c, e, l, p, m, n***	***f, c, a, m, p***

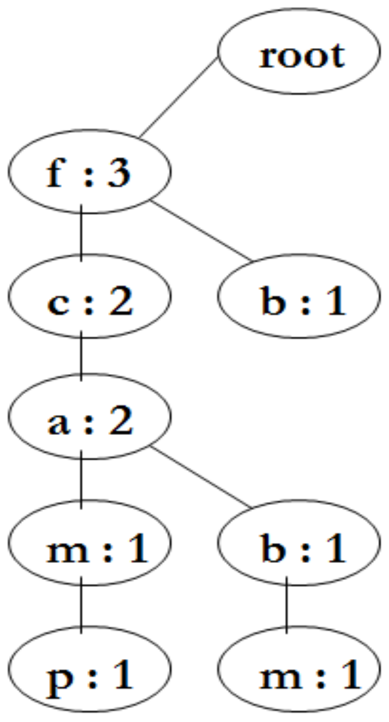
Now, all the Ordered-Item sets are inserted into a Trie Data Structure (frequent pattern tree).



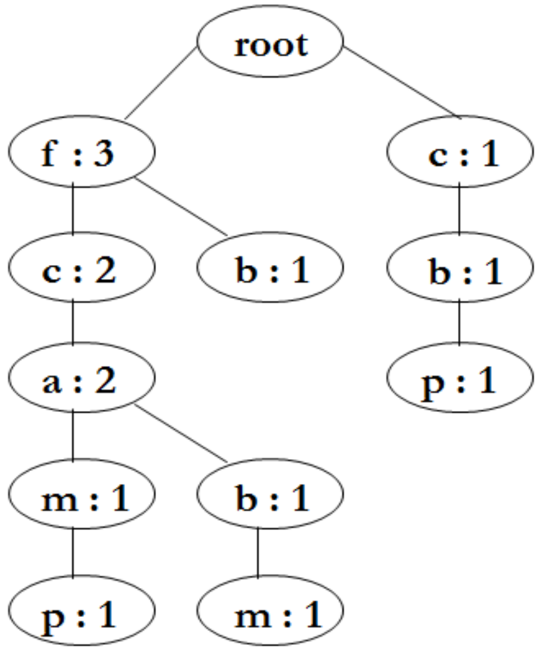
a) Inserting Ordered frequent items of TID-100



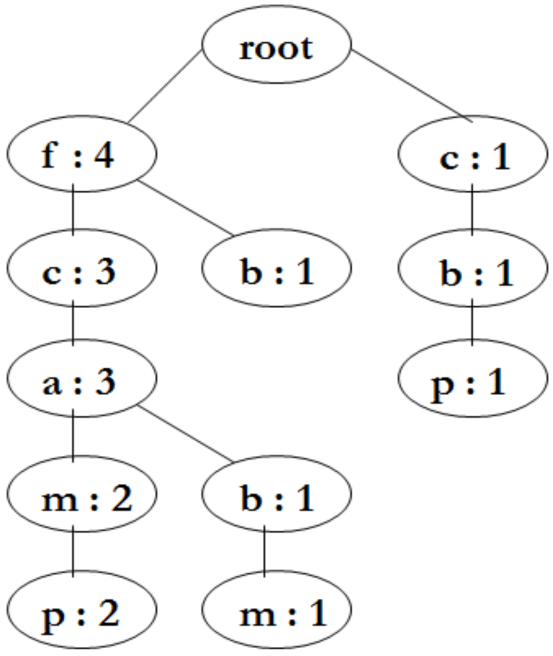
b) Inserting Ordered items of TID-200



c) Inserting Ordered items of TID-300



d) Inserting Ordered items of TID-400



e) Inserting Ordered items of TID-500

Now, for each item, the **Conditional Pattern Base** is computed which is the path labels of all the paths which lead to any node of the given item in the frequent-pattern tree.

Item	**Conditional Pattern Base**
p	{{f, c, a, m : 2}, {c, b : 1}}
m	{{f, c, a : 2}, {f, c, a, b : 1}}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}
a	{{f, c : 3}}
c	{{f : 3}}
f	Φ

Now for each item, the **Conditional Frequent Pattern Tree** is built. It is done by taking the set of elements that is common in all the paths in the Conditional Pattern Base of that item and calculating its support count by summing the support counts of all the paths in the Conditional Pattern Base.

Item	**Conditional Pattern Base**	**Conditional FP-Tree**
p	{{f, c, a, m : 2}, {c, b : 1}}	{c : 3}
m	{{f, c, a : 2}, {f, c, a, b : 1}}	{f, c, a :3}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}	Φ
a	{{f, c : 3}}	{f, c : 3}
c	{{f : 3}}	{f : 3}
f	Φ	Φ

From the Conditional Frequent Pattern tree, the Frequent Pattern rules are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding item.

Item	**Conditional Pattern Base**	**Conditional FP-Tree**	**Frequent Patterns Generated**
p	{{f, c, a, m : 2}, {c, b : 1}}	{c : 3}	{<c, p : 3>}
m	{{f, c, a : 2}, {f, c, a, b : 1}}	{f, c, a :3}	{ <f, m : 3>, <c, m : 3> <a, m : 3>, <f, c, m : 3> <f, a, m : 3>, <c, a, m :3>}
b	{{f, c, a : 1}, {f : 1}, {c : 1}}	Φ	{}
a	{{f, c : 3}}	{f, c : 3}	{<f, a : 3>, <c, a : 3>, <f, c, a:3>}
c	{{f : 3}}	{f : 3}	{ <f, c : 3>}
f	Φ	Φ	{}

For each row, two types of association rules can be inferred.

Chi-Square Test of Independence

A teacher wants to know if students' grades are related to their study habits (regular or irregular). A random sample of 120 students is summarized below:

Study Habits	Pass	Fail	Total
Regular	50	10	60
Irregular	30	30	60
Total	80	40	120

Null hypothesis (H_0): This is the "no effect," "no difference," or "nothing's going on" statement. It assumes that any observed change or association is due to random chance, not a real underlying relationship.
Example: "Study habits and exam results are independent."

Alternative hypothesis (H_1 or Ha): This claims the opposite — that something *is* happening. There's a real difference, association, or effect beyond random chance.
Example: "Study habits and exam results are not independent."

Hypotheses

- H_0: Study habits and exam results are independent.
- H_1: Study habits and exam results are not independent.

Expected frequencies (E = (row total × column total) / grand total)

- Regular–Pass: $(60\times80)/120 = 40$
- Regular–Fail: $(60\times40)/120 = 20$
- Irregular–Pass: $(60\times80)/120 = 40$
- Irregular–Fail: $(60\times40)/120 = 20$

Test statistic ($\chi^2 = \Sigma\,(O-E)^2 / E$)

- Regular–Pass: $(50-40)^2/40 = 2.5$
- Regular–Fail: $(10-20)^2/20 = 5.0$
- Irregular–Pass: $(30-40)^2/40 = 2.5$
- Irregular–Fail: $(30-20)^2/20 = 5.0$
- **Total χ^2 = 15.0**

Degrees of freedom
$df = (r-1)(c-1) = (2-1)(2-1) =$ **1**

Decision rule at $\alpha = 0.05$
Critical $\chi^2(\alpha=0.05, df=1)$ = **3.841**.
Since **15.0 > 3.841**, reject H_0.

Conclusion
There is a significant relationship between study habits and exam results. Students who study regularly are more likely to pass.

What is a Chi-Square Table?

A chi-square (χ^2) table lists **critical values** from the chi-square distribution for different **degrees of freedom (df)** and **right-tail significance levels (α)**. You use it to decide whether to reject H_0:

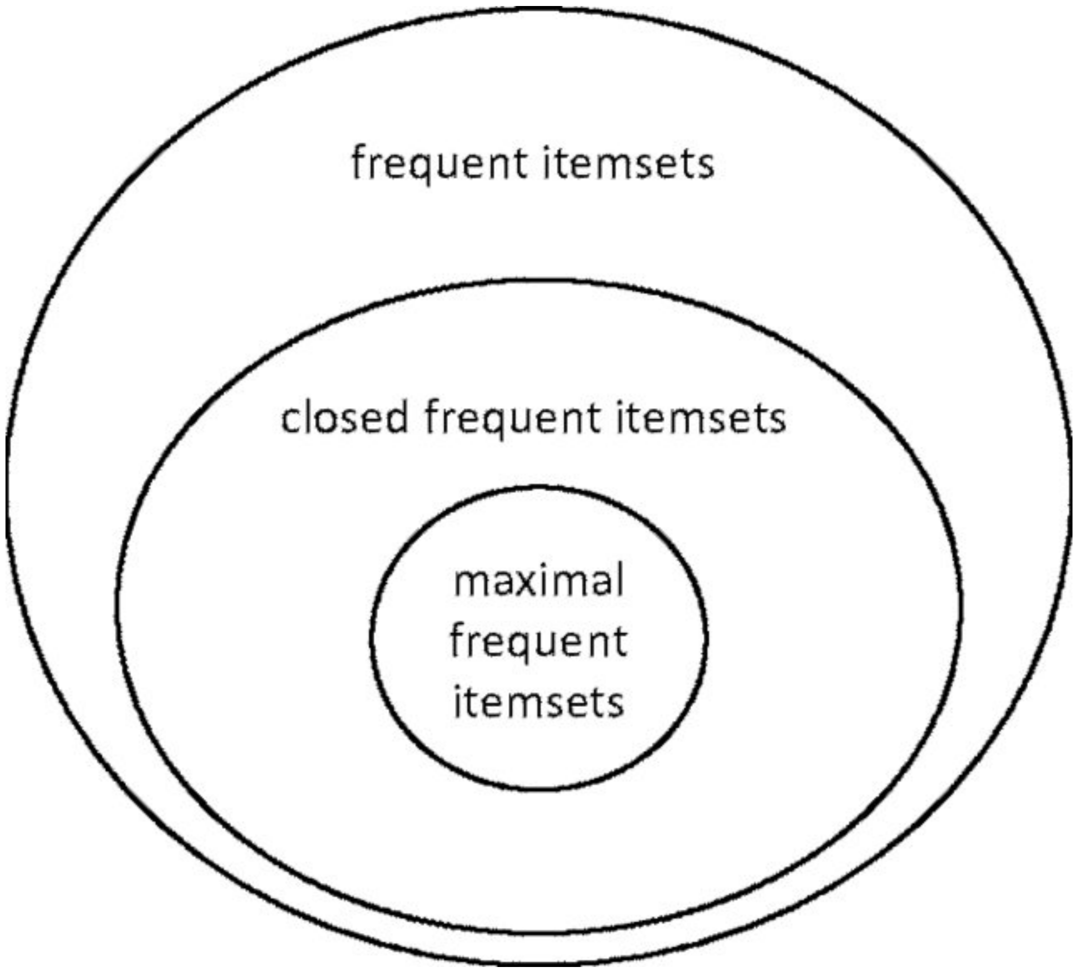
1. Compute your test statistic χ^2.
2. Find your df. For an r×c table, df = $(r-1)(c-1)$.
3. Choose α (commonly 0.10, 0.05, or 0.01).
4. Look up the **critical value** for your df and α.
5. If χ^2 (calculated) $\geq$ χ^2 (critical), **reject H_0**.

Compact χ^2 Critical Values (right-tail)

```
Right-tail critical values x²(α, df)

df |  α=0.10  α=0.05  α=0.01
---+-------------------------
 1 |   2.706   3.841   6.635
 2 |   4.605   5.991   9.210
 3 |   6.251   7.815  11.345
 4 |   7.779   9.488  13.277
 5 |   9.236  11.070  15.086
 6 |  10.645  12.592  16.812
 7 |  12.017  14.067  18.475
 8 |  13.362  15.507  20.090
 9 |  14.684  16.919  21.666
10 |  15.987  18.307  23.209
```

Reading tip: In the example, df=1 and α=0.05, so the critical value is 3.841. Because 15.0 ≥ 3.841, H_0 is rejected.


frequent itemsets
closed frequent itemsets
maximal frequent itemsets

Mining Closed and Max Patterns (Easily Explained)

When mining frequent itemsets, we often get **too many** results, which can be overwhelming. To solve this, we focus on **closed** and **maximal** frequent itemsets, which **reduce redundancy** while still keeping useful information.

What Are Closed and Maximal Frequent Itemsets?

✅ **Closed Frequent Itemset:**
A frequent itemset is **closed** if **no larger itemset** has the **same support** as it.
🔹 **Example:** If `{A, B, C}` appears in 5 transactions, but `{A, B}` also appears in 5 transactions, `{A, B}` is **not closed because** adding `C` did not reduce its support. `{A, B, C}` is closed.

✅ **Maximal Frequent Itemset:**
A frequent itemset is **maximal** if **no larger frequent itemset** exists.
🔹 **Example:** If `{A, B, C}` is frequent but `{A, B, C, D}` is **not**, then `{A, B, C}` is **maximal**.

👉 **Closed sets keep more information than maximal sets!**

Suppose the **minimum support** is 2.
Any bucket with a total count less than 2 can't possibly contain frequent itemsets — so we drop pairs from those buckets.

That means we only keep pairs from buckets **2, 5, and 6**, and ignore the rest.

So `{A, B}`, `{A, C}`, and `{B, C}` survive to the next stage, while `{A, D}` and `{B, D}` are discarded early.

When scanning the dataset, instead of counting every pair individually, we **increment the count for its bucket** whenever that pair appears.

Let's say the results after scanning all transactions are:

Bucket	Count
0	1
2	2
3	1
5	2
6	2

Assume A=1, B=2, C=3, D=4.

Now hash the pairs:

Pair	Hash Calculation	Bucket
{A, B}	(1×10 + 2) mod 7 = 12 mod 7 = 5	5
{A, C}	(1×10 + 3) mod 7 = 13 mod 7 = 6	6
{B, C}	(2×10 + 3) mod 7 = 23 mod 7 = 2	2
{A, D}	(1×10 + 4) mod 7 = 14 mod 7 = 0	0
{B, D}	(2×10 + 4) mod 7 = 24 mod 7 = 3	3

Now each pair "lands" in a bucket.

Transaction ID	Items Bought
T1	{A, B, C}
T2	{A, C}
T3	{B, C}
T4	{A, B, D}

We want to find **frequent 2-itemsets** (pairs of items that often occur together).