CST466
DATA MINING

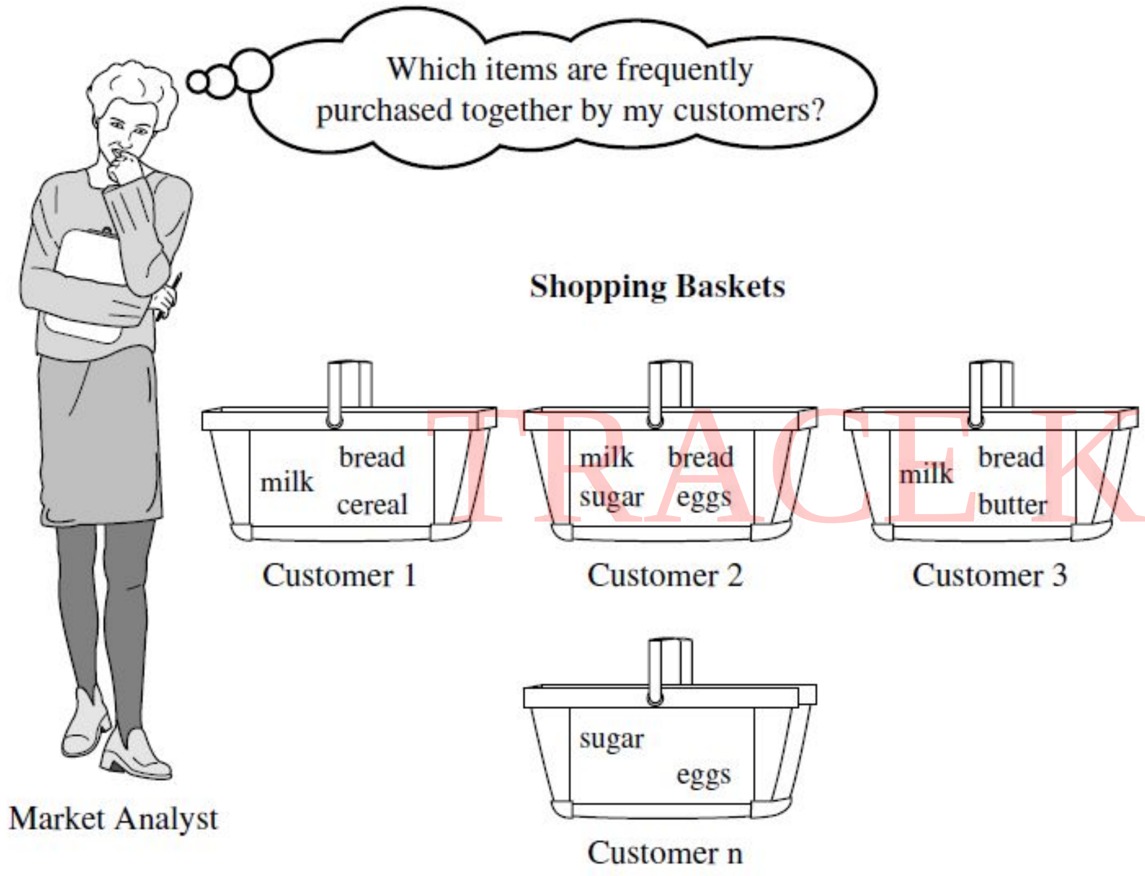
MODULE-4

Module 4: (Association Rule Analysis)

Association Rules-Introduction, Methods to discover Association rules, Apriori(Level-wise algorithm), Partition Algorithm, Pincer Search Algorithm, Dynamic Itemset Counting Algorithm, FP-tree Growth Algorithm.

ASSOCIATION RULE ANALYSIS:

- Association rule mining **finds interesting associations and relationships among large sets of data items.**
- This rule shows how frequently an itemset occurs in a transaction.
- A typical example is a **Market Based Analysis.**
    - It allows retailers to identify relationships between the items that people buy together frequently.
    - This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.
    - The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.



Eg:

- If customers are buying milk, how likely are they to also buy bread on the same trip to the supermarket?
- Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

Association Analysis : Basic Concepts

Frequent Itemset

- A set of items is referred to as an itemset.
- An itemset that contains k items is a k-itemset.
- The set {computer, antivirus_software} is a 2-itemset.
- **A frequent item set is a set of items that occur together frequently in a dataset.**
- Eg1: In a supermarket environment, the items bread and butter are likely to be purchased together by many customers.
  - So, {bread, butter} is an example for frequent itemset.
  - The association between the items are represented by the association rule;

bread=>butter

- Eg2: In an electronic store, customers who purchase computers also tend to buy antivirus software at the same time.
  - So, {computer,antivirus software} is an example for frequent itemset.
  - It is represented by the following association rule;

$$computer \Rightarrow antivirus\_software$$

<u>Measures of Rule Interestingness:</u>

- Support and Confidence are two measures of rule interestingness.
- **Support** reflects the **usefulness** of discovered association rules.
- **Confidence** reflects the **certainty** of discovered association rules.
- Association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.
    - Such thresholds can be set by users or domain experts.
- <u>Eg:</u>

Consider the following association rule;

$$computer \Rightarrow antivirus\_software \ [support = 2\%, confidence = 60\%]$$

- A support of 2% for Association rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.

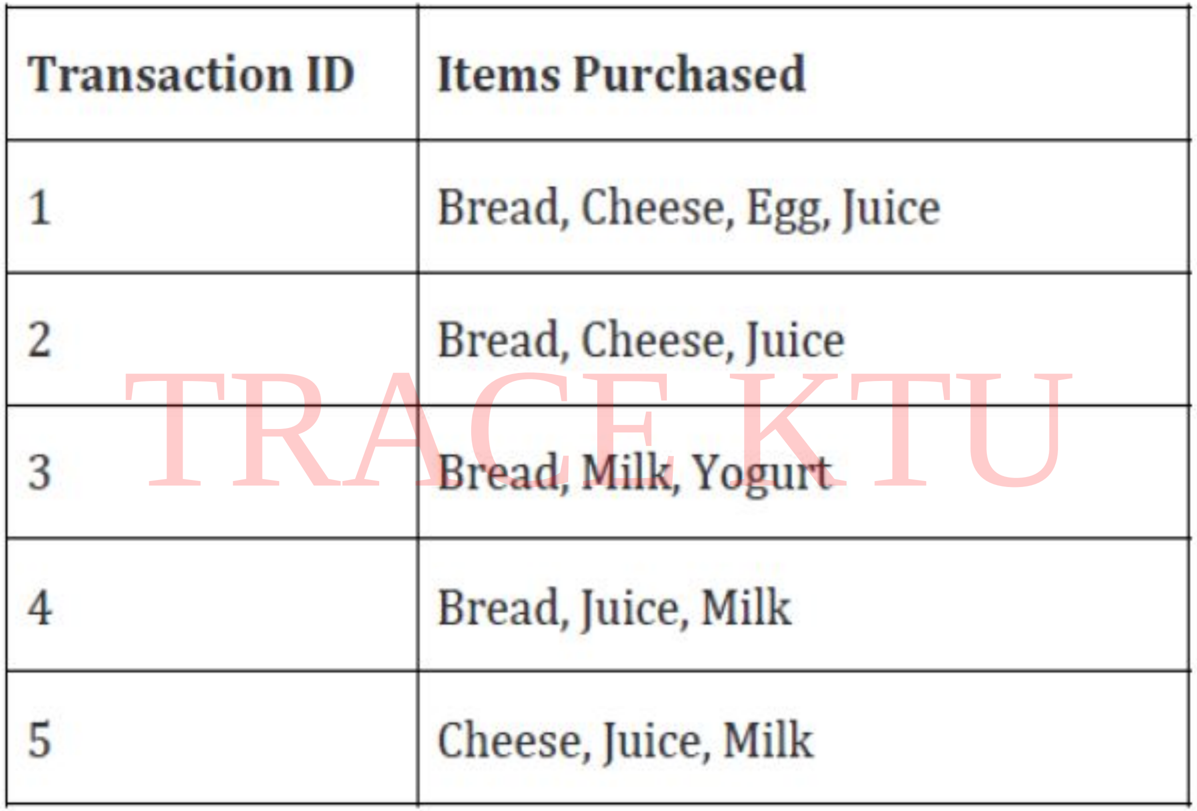
$$
\begin{aligned}
support(A \Rightarrow B) &= P(A \cup B) \\
confidence(A \Rightarrow B) &= P(B|A).
\end{aligned}
$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

- In general, **association rule mining** can be viewed as a **two-step process:**

1. **Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min_sup*.
2. **Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

Q:Find the support of each item in the dataset.



Transaction ID	Items Purchased
1	Bread, Cheese, Egg, Juice
2	Bread, Cheese, Juice
3	Bread, Milk, Yogurt
4	Bread, Juice, Milk
5	Cheese, Juice, Milk

Support (item) = Frequency of item/Number of transactions

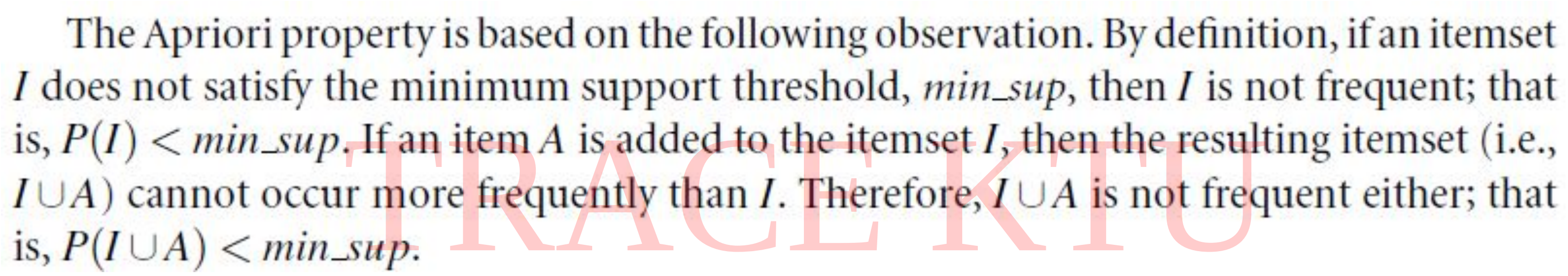
Item	Frequency	Support (in %)
Bread	4	4/5=80%
Cheese	3	3/5=60%
Egg	1	1/5=20%
Juice	4	4/5=80%
Milk	3	3/5=60%
Yogurt	1	1/5=20%

The Apriori Algorithm: (Level wise algorithm) ***

- Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant.
- Also known as **level-wise algorithm.**
- Used for **mining frequent itemsets for Boolean association rules.**
- The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties.
- Apriori employs an **iterative approach** known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets.
  - First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support.
    - The resulting set is denoted L1.
  - Next, L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found.
  - The finding of each L_k requires one full scan of the database.

- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property** is used to reduce the search space.

Apriori property: All nonempty subsets of a frequent itemset must also be frequent.



The Apriori property is based on the following observation. By definition, if an itemset I does not satisfy the minimum support threshold, *min_sup*, then I is not frequent; that is, $P(I) < min\_sup$. If an item A is added to the itemset I, then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I. Therefore, $I \cup A$ is not frequent either; that is, $P(I \cup A) < min\_sup$.

- This property belongs to a special category of properties called **antimonotone** in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well.
- It is called antimonotone because the property is monotonic in the context of failing a test.

“How is the Apriori property used in the algorithm?”

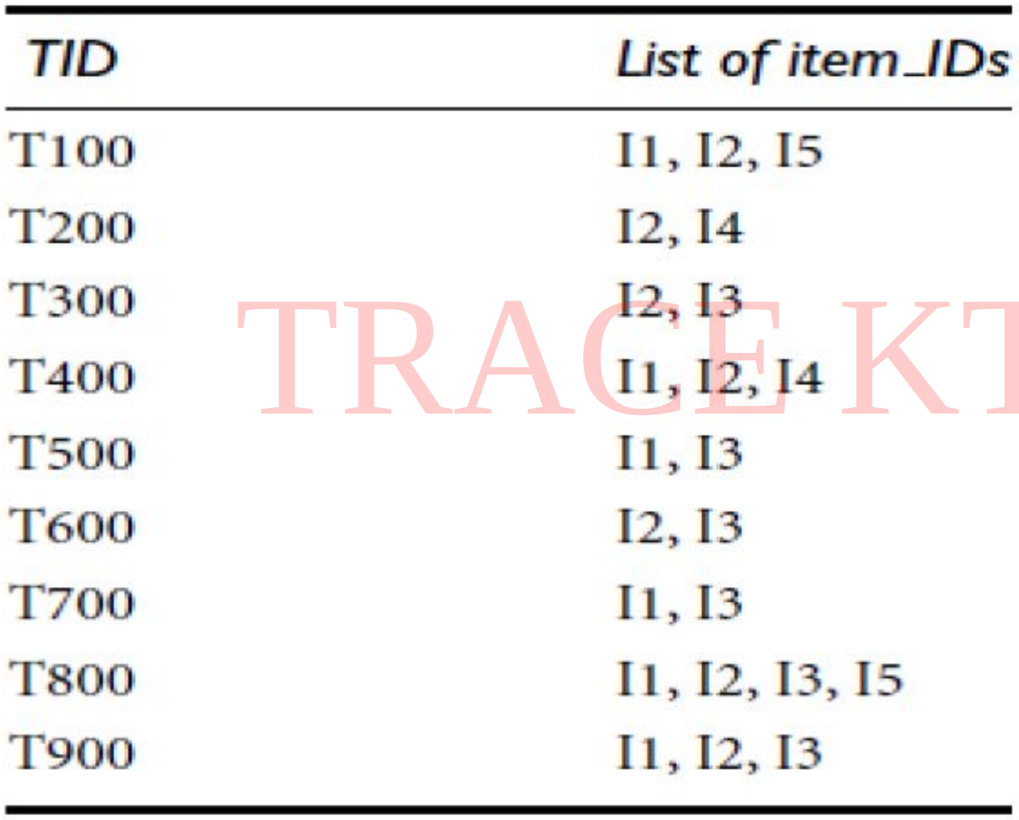
1. The join step:

- To find L_k, a set of candidate k-itemsets is generated by joining L_{k-1} with itself.
- This set of candidates is denoted C_k.

2. The prune step:

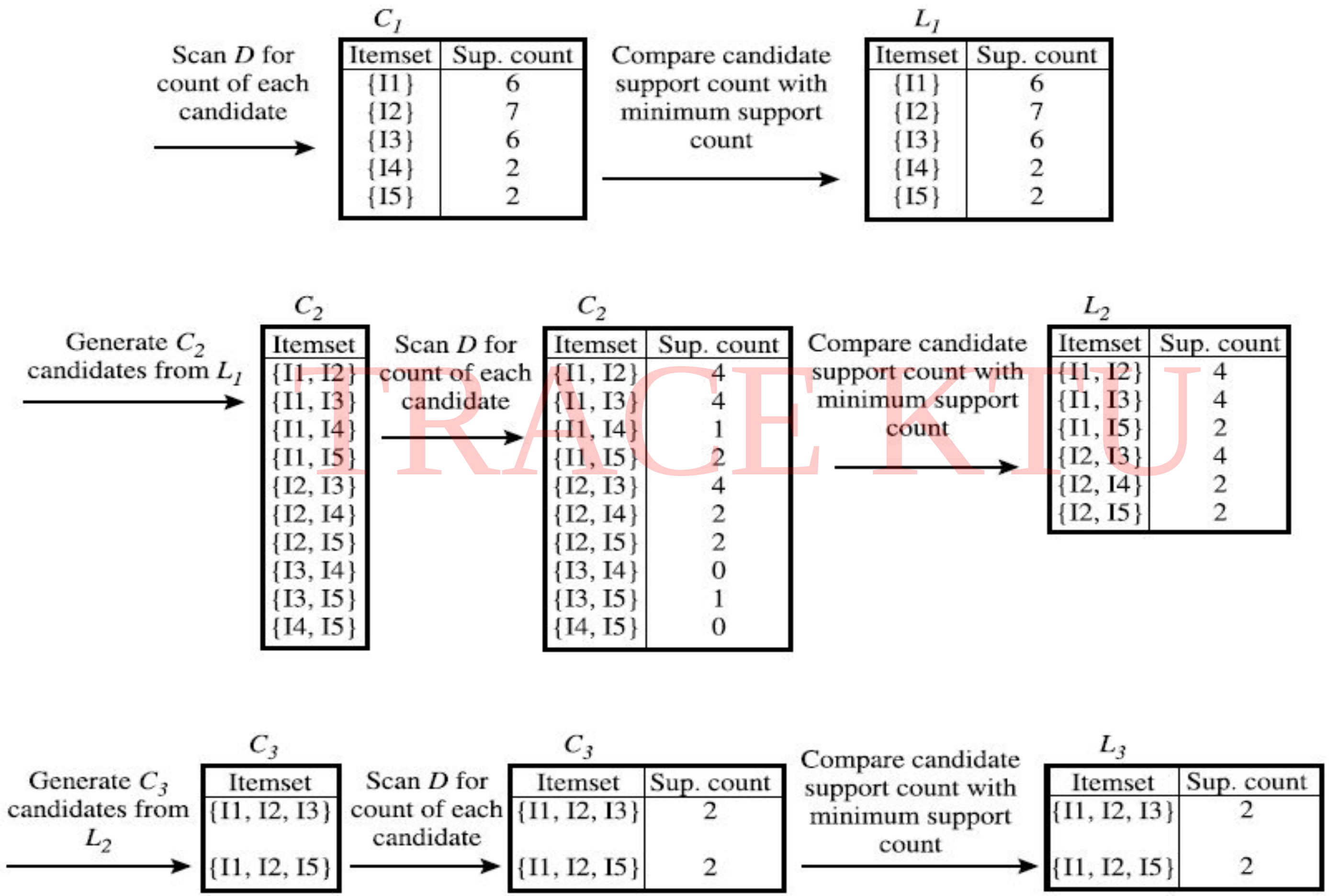
- A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk)
- C_k, however, can be huge, and so this could involve heavy computation.
- To reduce the size of C_k, the Apriori property is used as follows.
  - Any (k□1)-itemset that is not frequent cannot be a subset of a frequent k-itemset.
  - Hence, if any (k-1)-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k.

Q: Find Association rules from the following data using Apriori algorithm with minimum support count required is 2 and confidence 70%



TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

<u>Step 1:</u> Finding frequent itemset:



Scan *D* for count of each candidate →

C₁

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count →

L₁

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 →

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan *D* for count of each candidate →

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count →

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan *D* for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Step 2: Generation of strong association rules from the frequent-itemset.

$$\begin{array}{ll}
I1 \wedge I2 \Rightarrow I5, & confidence = 2/4 = 50\% \\
I1 \wedge I5 \Rightarrow I2, & confidence = 2/2 = 100\% \\
I2 \wedge I5 \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow I2 \wedge I5, & confidence = 2/6 = 33\% \\
I2 \Rightarrow I1 \wedge I5, & confidence = 2/7 = 29\% \\
I5 \Rightarrow I1 \wedge I2, & confidence = 2/2 = 100\%
\end{array}$$

- Given : The minimum confidence threshold is 70%.
- So, only the second, third, and last rules are strong association rules.

Explanation to the previous example:

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.
2. Suppose that the minimum support count required is 2, that is, $min\_sup = 2$. (Here, we are referring to *absolute* support because we are using a support count. The corresponding relative support is 2/9 = 22%). The set of frequent 1-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. In our example, all of the candidates in C_1 satisfy minimum support.
3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses the join $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C_2.[8] C_2 consists of $\binom{|L_1|}{2}$ 2-itemsets. Note that no candidates are removed from C_2 during the prune step because each subset of the candidates is also frequent.
4. Next, the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in Figure 5.2.
5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

6. The generation of the set of candidate 3-itemsets, C_3, is detailed in Figure 5.3. From the join step, we first get $C_3 = L_2 \bowtie L_2 = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3. Note that when given a candidate k-itemset, we only need to check if its $(k-1)$-subsets are frequent since the Apriori algorithm uses a level-wise search strategy. The resulting pruned version of C_3 is shown in the first table of the bottom row of Figure 5.2.

7. The transactions in D are scanned in order to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support (Figure 5.2).

8. The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in $\{\{I1, I2, I3, I5\}\}$, this itemset is pruned because its subset $\{\{I2, I3, I5\}\}$ is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets. ■

How can we further improve the efficiency of Apriori-based mining?

1. **Use hash-based techniques:**
   - A hash-based technique can be used to reduce the size of the candidate k-itemsets, Ck, for k > 1.
2. **Transaction reduction:**
   - Reducing the number of transactions scanned in future iterations.
   - A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k+1)-itemsets.
   - Therefore, such a transaction can be marked or removed from further consideration
3. **Partitioning ✔**
   - Partitioning the data to find candidate itemsets.
   - A partitioning technique can be used that requires just two database scans to mine the frequent itemsets.
4. **Dynamic itemset counting ✔**
   - Adding candidate itemsets at different points during a scan.
5. **Sampling**