CM2604 Machine Learning

Association Mining

Week 07 | Prasan Yapa

Content

- What is association rule mining
- Frequent itemsets, support, and confidence
- Mining association rules
- Apriori algorithm
- Rule generation

Association Rule Learning

Association Rule Learning

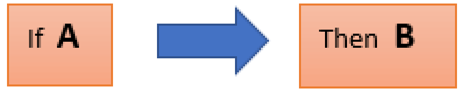
- Association rule learning is a type of unsupervised learning technique.
- It checks for the dependency of one data item on another data item and maps accordingly.
- It is based on different rules to discover the interesting relations between variables in the database.
- It is employed in **Market Basket analysis, Web usage mining, continuous production**, etc.

Question

How can we mine **interesting patterns** and **useful rules** from data?

How does Association Rule Learning Work?

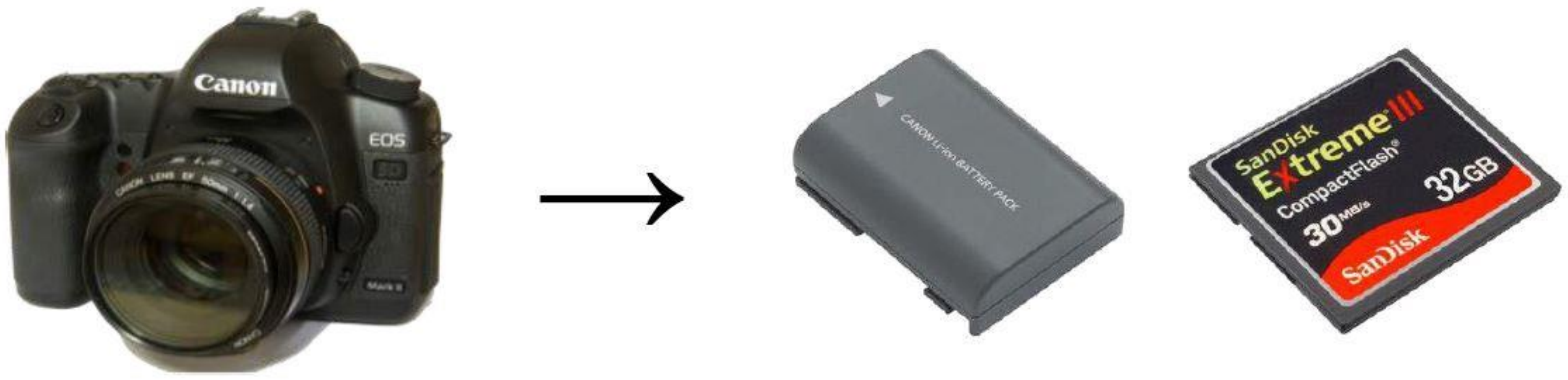
- Association rule learning works on the concept of If and Else Statement, such as if A then B.



- Here the If element is called antecedent, and then statement is called as Consequent.
- These types of relationships where we can find out some association or relation between two items is known as single cardinality.
- It is all about creating rules, and if the number of items increases, then cardinality also increases.

Example

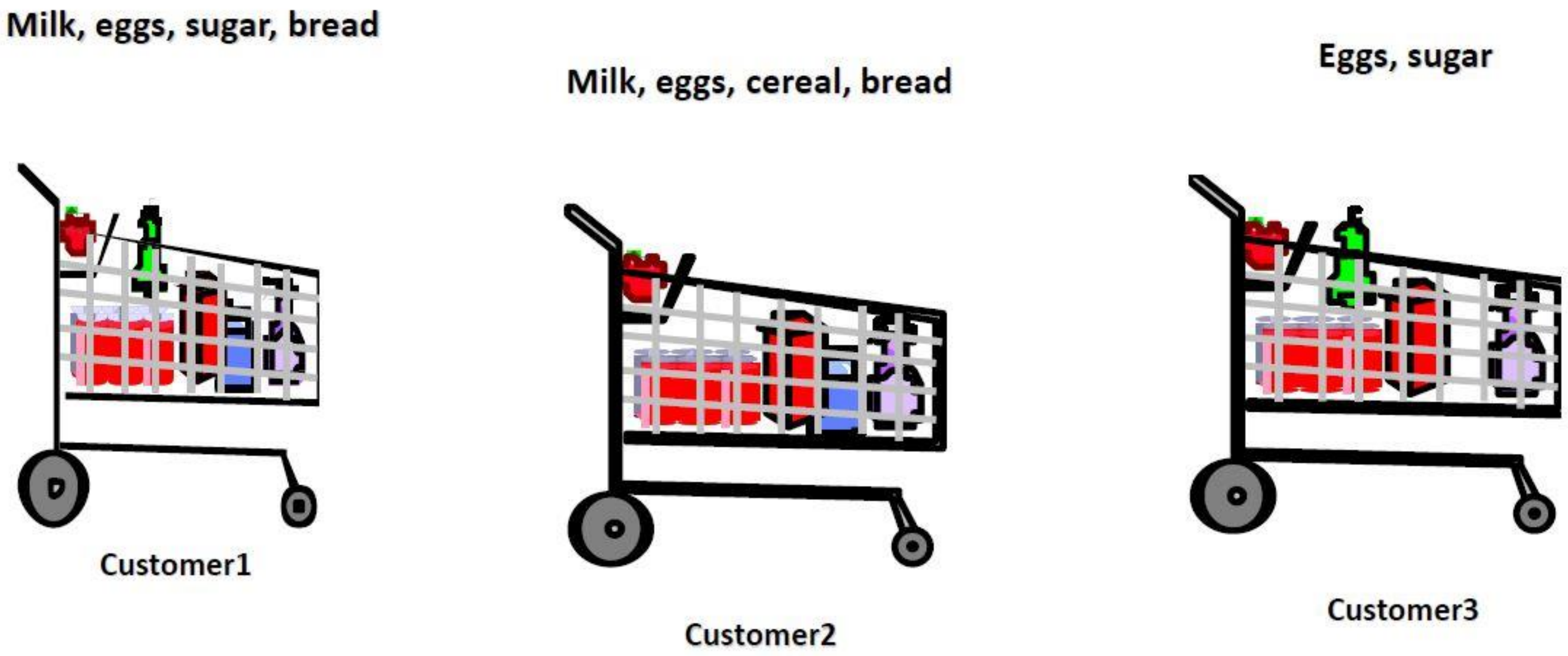
You run an on-line store, and want to increase sales. You decide on **associative advertising**: show ads of relevant products **before** your users search for these



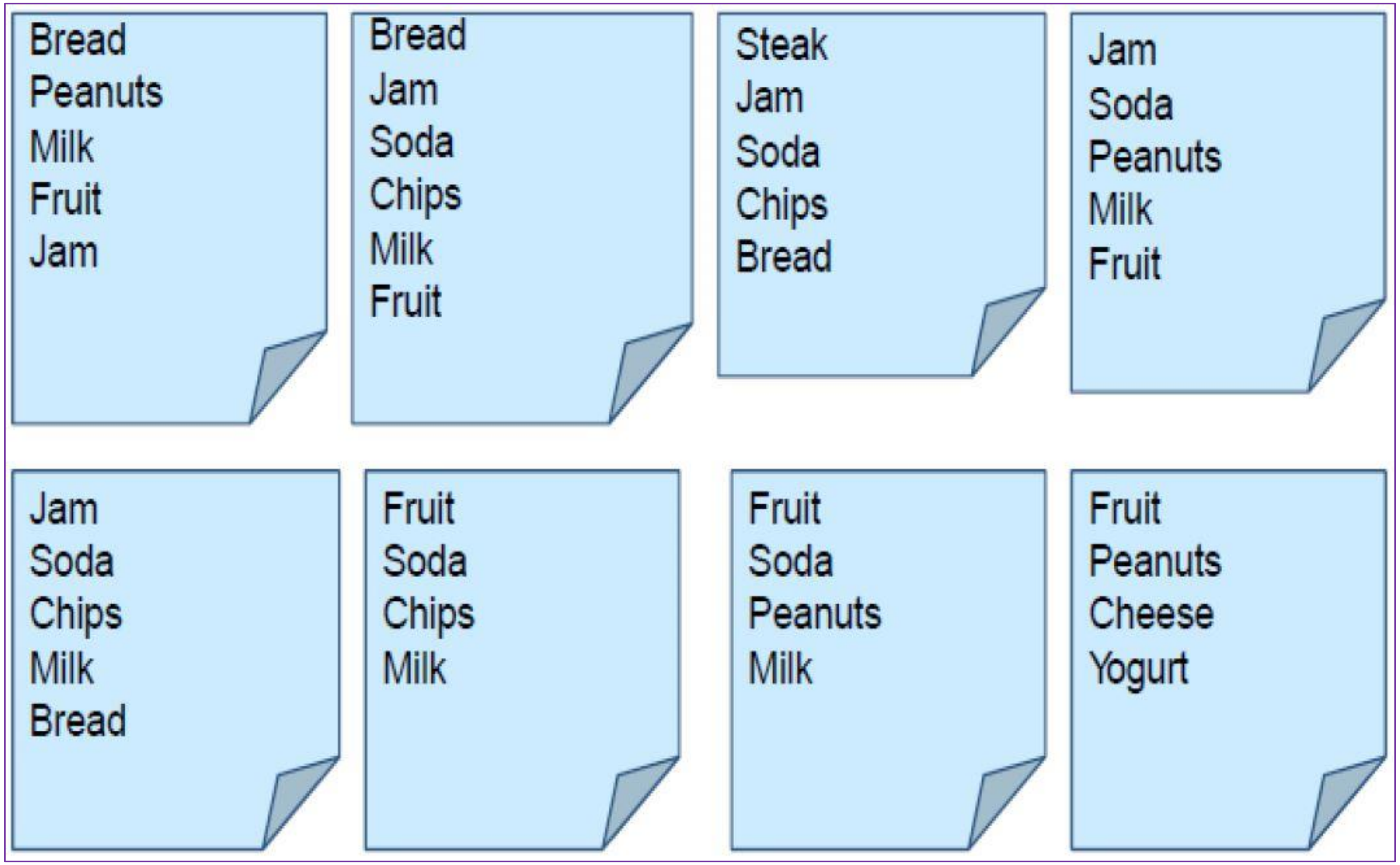
Easy, knowing the left-hand side. What if we don't?

Market Basket Analysis

- Analysis of customer buying habits by finding associations and correlations between the different items that customers place in their "shopping basket"



Market Basket Analysis Transactions



Bread
Peanuts
Milk
Fruit
Jam

Bread
Jam
Soda
Chips
Milk
Fruit

Steak
Jam
Soda
Chips
Bread

Jam
Soda
Peanuts
Milk
Fruit

Jam
Soda
Chips
Milk
Bread

Fruit
Soda
Chips
Milk

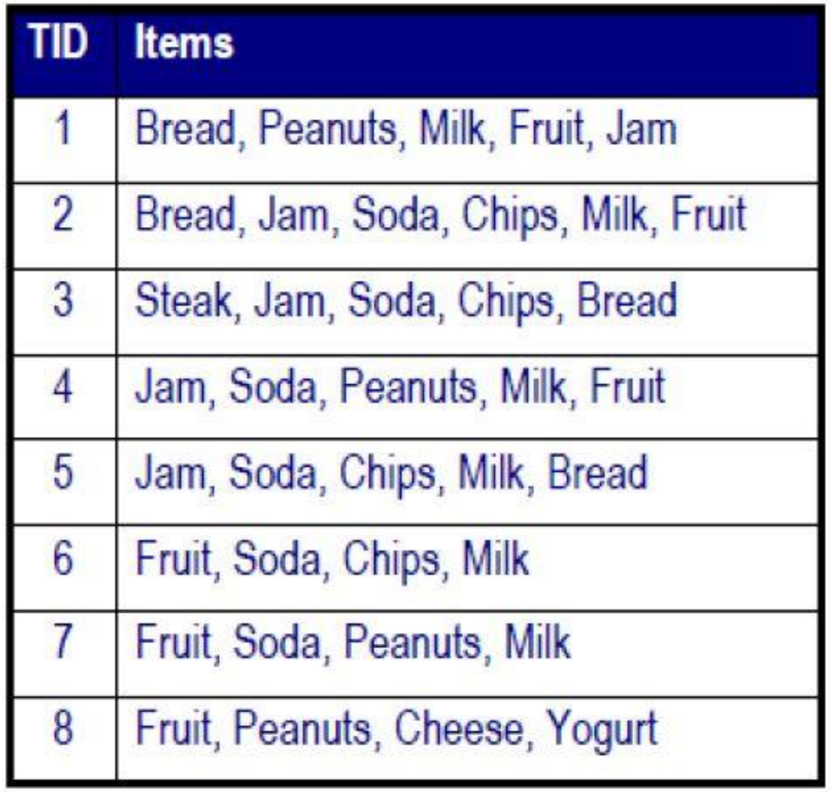
Fruit
Soda
Peanuts
Milk

Fruit
Peanuts
Cheese
Yogurt

Association Mining

- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- “IF buys beer and sausage, THEN also buy mustard with high probability”.
- “On Thursdays, grocery store consumers often purchase diapers and beer together”.
- “Customers who purchase maintenance agreements are very likely to purchase large appliances”.

What is Association Rule Mining?



TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Examples

$\{bread\} \Rightarrow \{milk\}$

$\{soda\} \Rightarrow \{chips\}$

$\{bread\} \Rightarrow \{jam\}$

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Support, Confidence & Lift

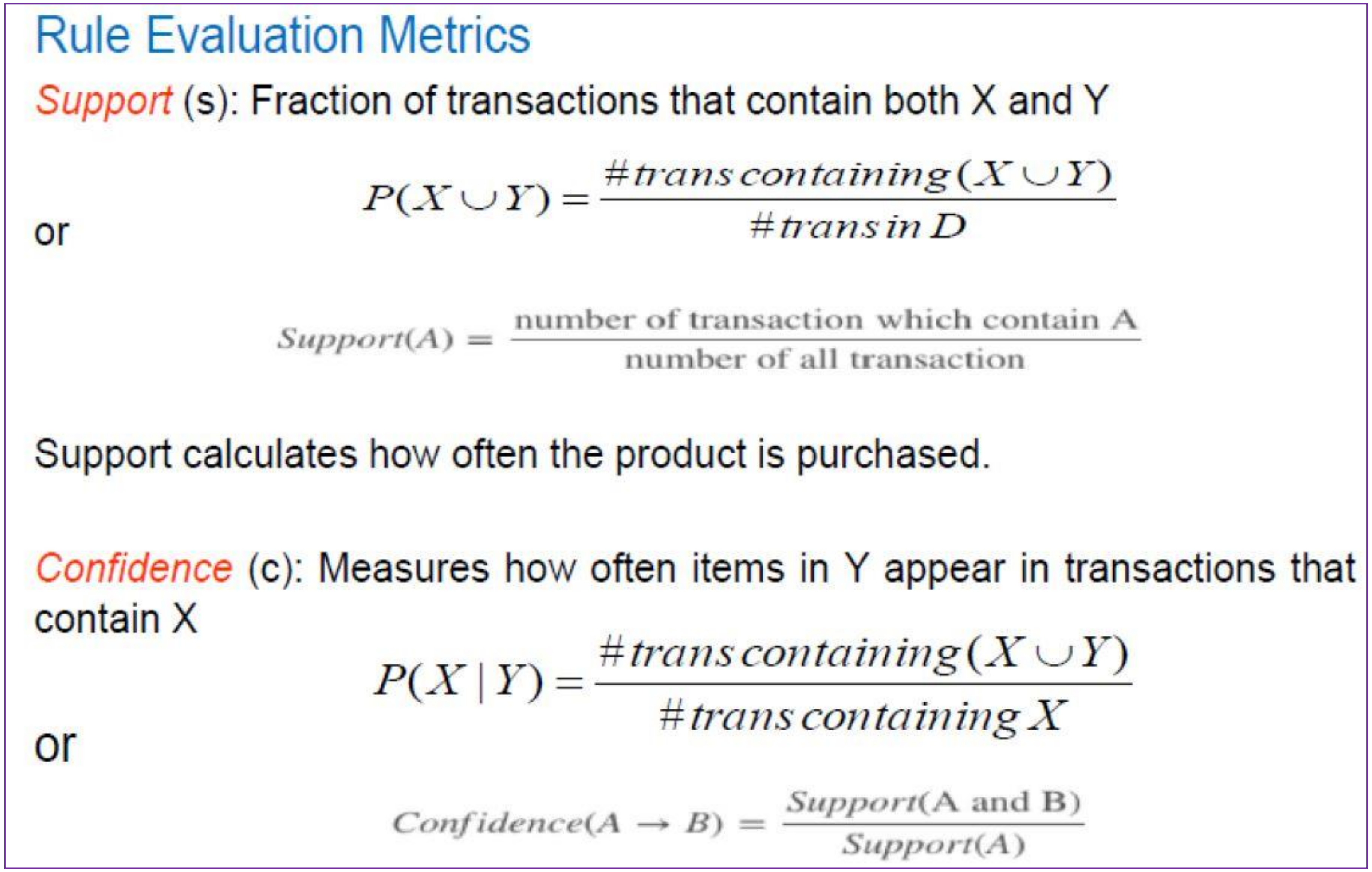
Metrics for Association Rules

- Support
  - Support is how frequently an item appears in the dataset.
  - It is defined as the fraction of the transaction T that contains the itemset X.
  - If there are X datasets, then for transactions T, it can be written as:
  - $Support(X) = \frac{Freq(X)}{T}$
- Confidence
  - Confidence indicates how often the rule has been found to be true.
  - Or how often the items X and Y occur together in the dataset when the occurrence of X is already given.
  - It is the ratio of the transaction that contains X and Y to the number of records that contain X.

Metrics for Association Rules

- Confidence
  - $Confidence = \frac{Freq(X,Y)}{Freq(X)}$
- Lift
  - It is the strength of any rule, which can be defined as below:
  - $Lift = \frac{Support(X,Y)}{Support(X) * Support(Y)}$
  - It is the ratio of the observed support measure and expected support.
  - *Lift=1:* The probability of antecedent and consequent is independent of each other.
  - *Lift>1:* The two itemsets are dependent to each other.
  - *Lift<1:* One item has a negative effect on another.

Metrics for Association Rules



Rule Evaluation Metrics

Support (s): Fraction of transactions that contain both X and Y

$$P(X \cup Y) = \frac{\#trans\ containing\ (X \cup Y)}{\#trans\ in\ D}$$

or

$$Support(A) = \frac{\text{number of transaction which contain A}}{\text{number of all transaction}}$$

Support calculates how often the product is purchased.

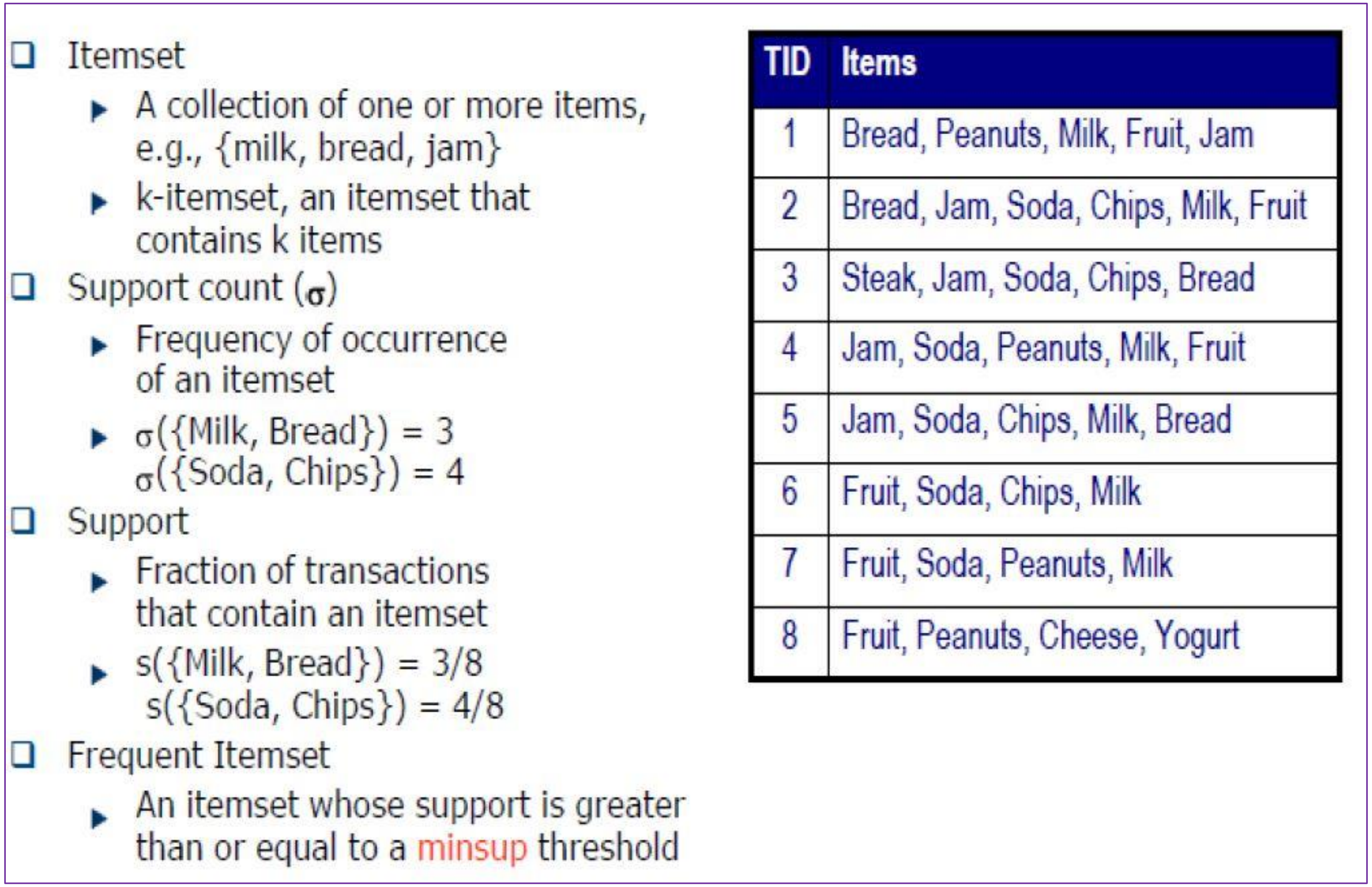
Confidence (c): Measures how often items in Y appear in transactions that contain X

$$P(X \mid Y) = \frac{\#trans\ containing\ (X \cup Y)}{\#trans\ containing\ X}$$

or

$$Confidence(A \rightarrow B) = \frac{Support(\text{A and B})}{Support(A)}$$

Frequent Itemset



- Itemset
  - A collection of one or more items, e.g., {milk, bread, jam}
  - k-itemset, an itemset that contains k items
- Support count (σ)
  - Frequency of occurrence of an itemset
  - σ({Milk, Bread}) = 3
    σ({Soda, Chips}) = 4
- Support
  - Fraction of transactions that contain an itemset
  - s({Milk, Bread}) = 3/8
    s({Soda, Chips}) = 4/8
- Frequent Itemset
  - An itemset whose support is greater than or equal to a minsup threshold

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

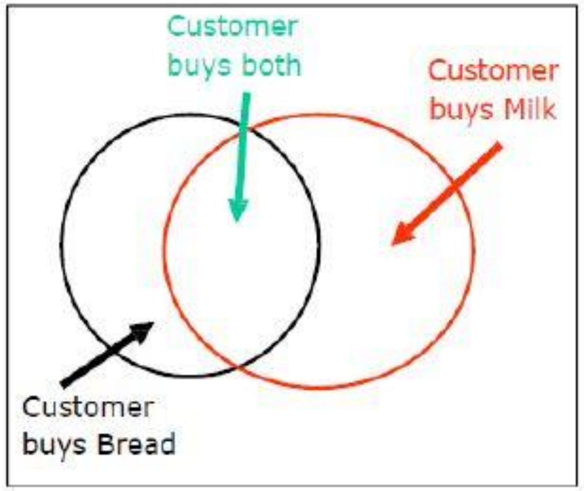
What is an Association Rule

- Implication of the form X $\Rightarrow$ Y, where X and Y are itemsets
- Example, {bread} $\Rightarrow$ {milk}
- Rule Evaluation Metrics, Suppor & Confidence
- Support (s)
  - Fraction of transactions that contain both X and Y

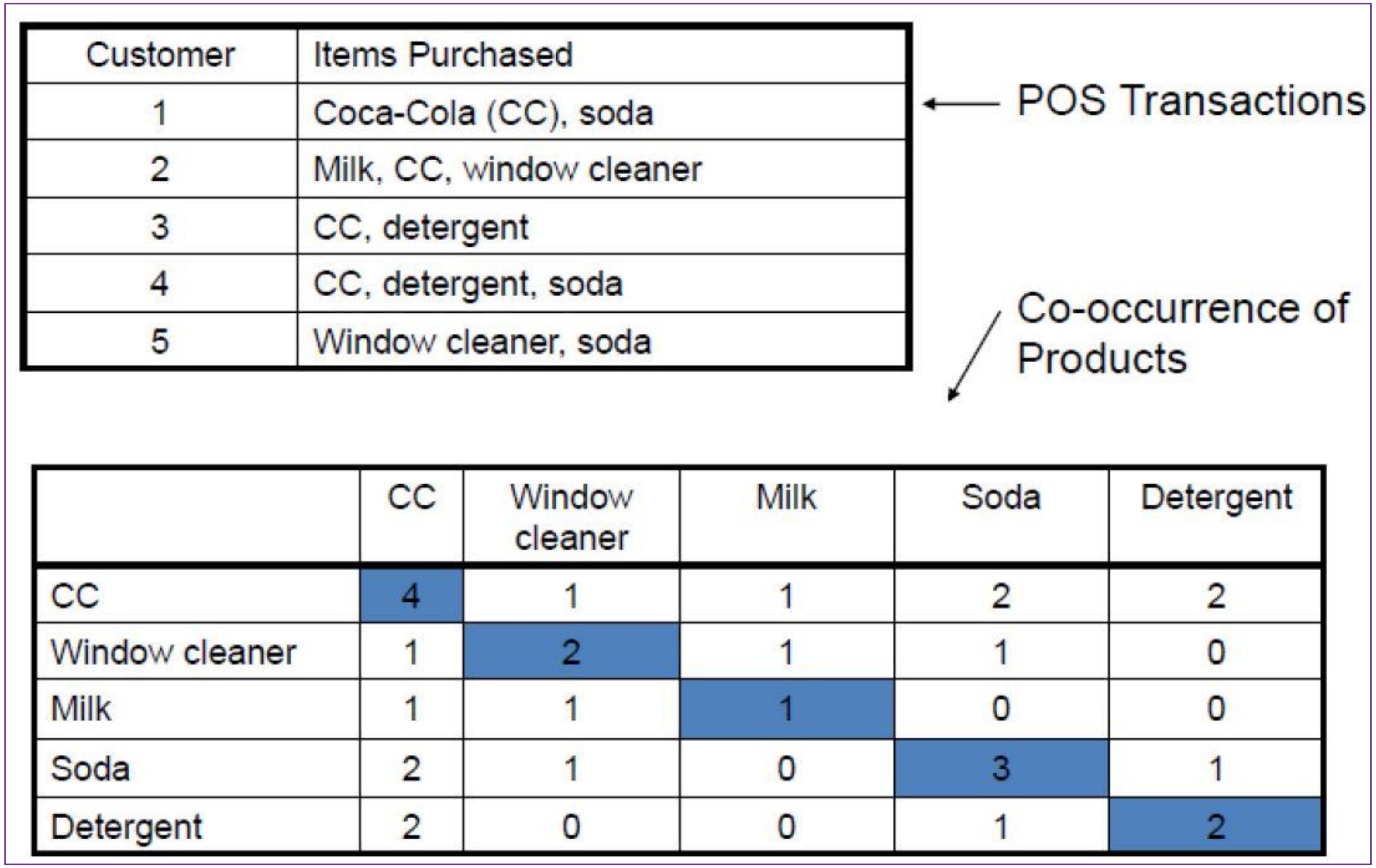
$$s = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\#\ \text{of transactions}} = 0.38$$

- Confidence (c)
  - Measures how often items in Y appear in transactions that contain X

$$c = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$



How Good is an Association Rule?



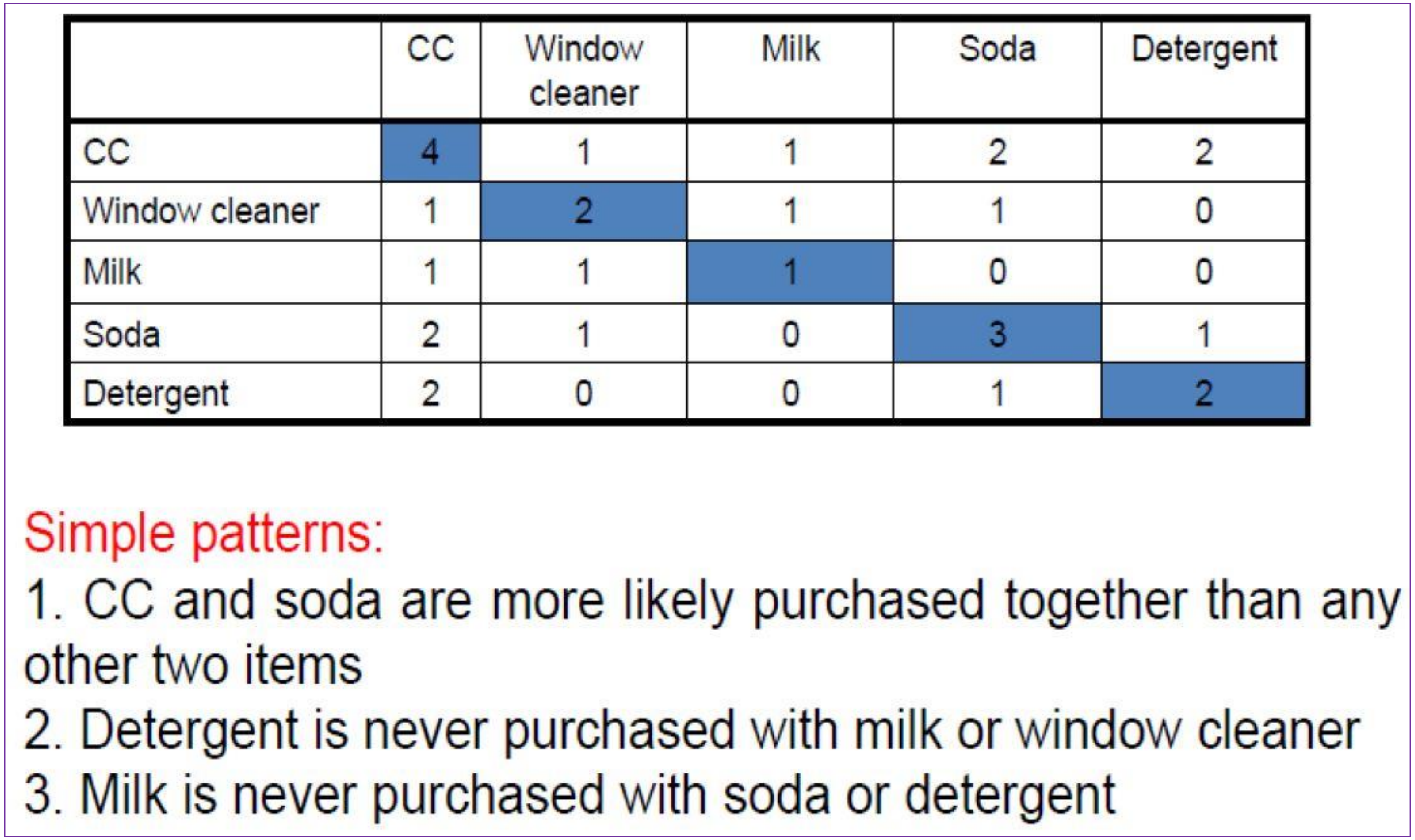
Customer	Items Purchased
1	Coca-Cola (CC), soda
2	Milk, CC, window cleaner
3	CC, detergent
4	CC, detergent, soda
5	Window cleaner, soda

POS Transactions

Co-occurrence of Products

	CC	Window cleaner	Milk	Soda	Detergent
CC	4	1	1	2	2
Window cleaner	1	2	1	1	0
Milk	1	1	1	0	0
Soda	2	1	0	3	1
Detergent	2	0	0	1	2

How Good is an Association Rule?

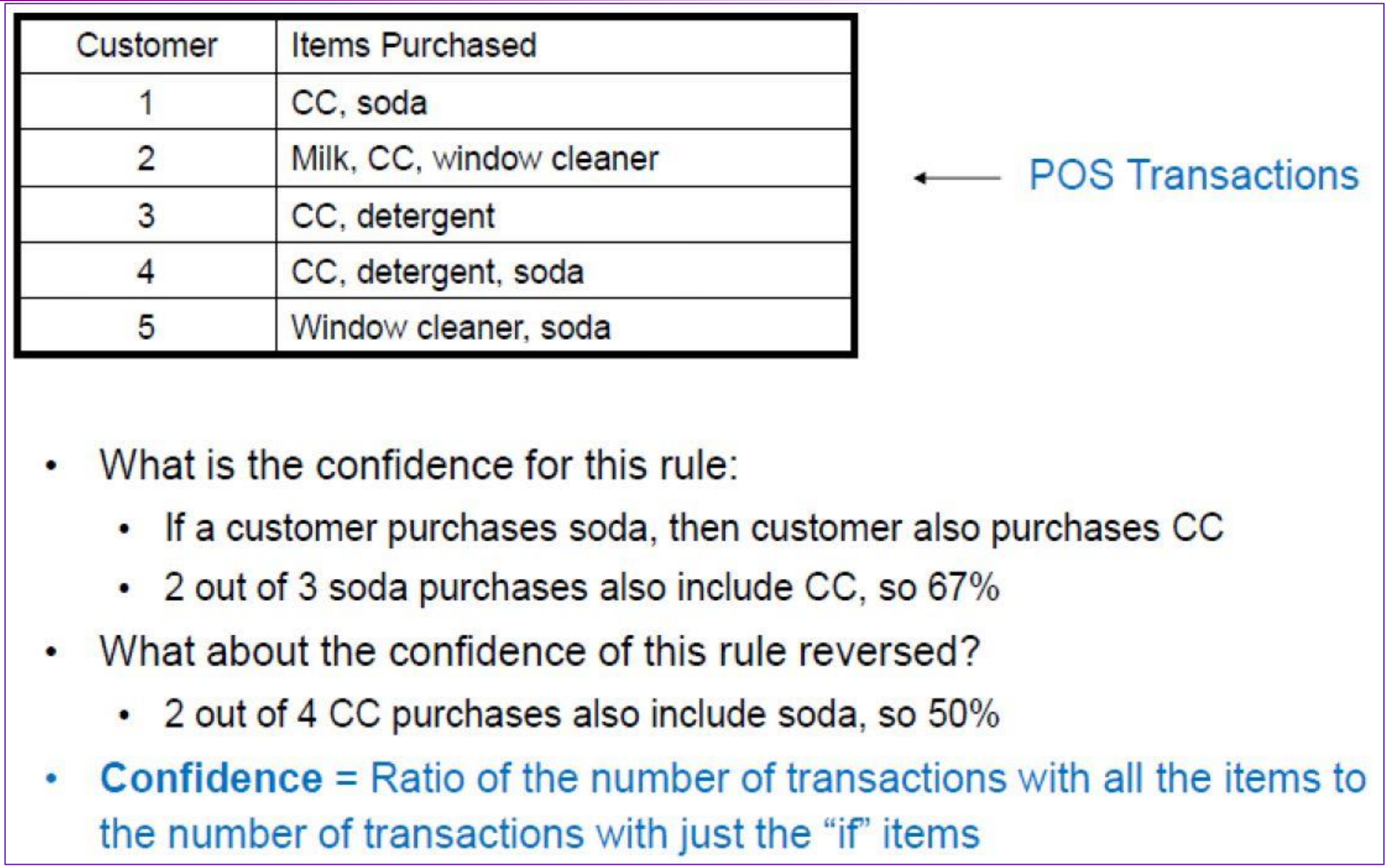


	CC	Window cleaner	Milk	Soda	Detergent
CC	4	1	1	2	2
Window cleaner	1	2	1	1	0
Milk	1	1	1	0	0
Soda	2	1	0	3	1
Detergent	2	0	0	1	2

Simple patterns:

1. CC and soda are more likely purchased together than any other two items
2. Detergent is never purchased with milk or window cleaner
3. Milk is never purchased with soda or detergent

How Good is an Association Rule?



Customer	Items Purchased
1	CC, soda
2	Milk, CC, window cleaner
3	CC, detergent
4	CC, detergent, soda
5	Window cleaner, soda

← POS Transactions

- What is the confidence for this rule:
  - If a customer purchases soda, then customer also purchases CC
  - 2 out of 3 soda purchases also include CC, so 67%
- What about the confidence of this rule reversed?
  - 2 out of 4 CC purchases also include soda, so 50%
- **Confidence** = Ratio of the number of transactions with all the items to the number of transactions with just the “if” items

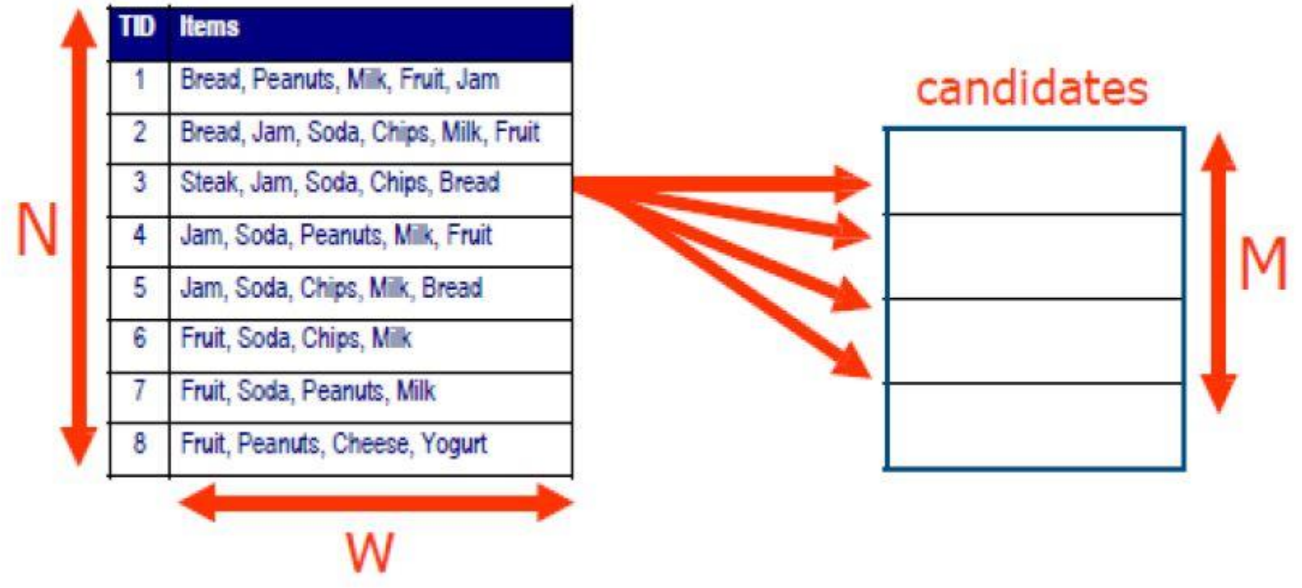
Mining Association Rules

Mining Association Rules

- Two step approach
  - Step 1 -> Frequent Itemset Generation
    - Generate all itemsets whose support >= **minsup**
  - Step 2 -> Rule Generation
    - Generate high confidence rules from each frequent itemset where each rule is a binary partitioning of a frequent itemset

Frequent Itemset Generation

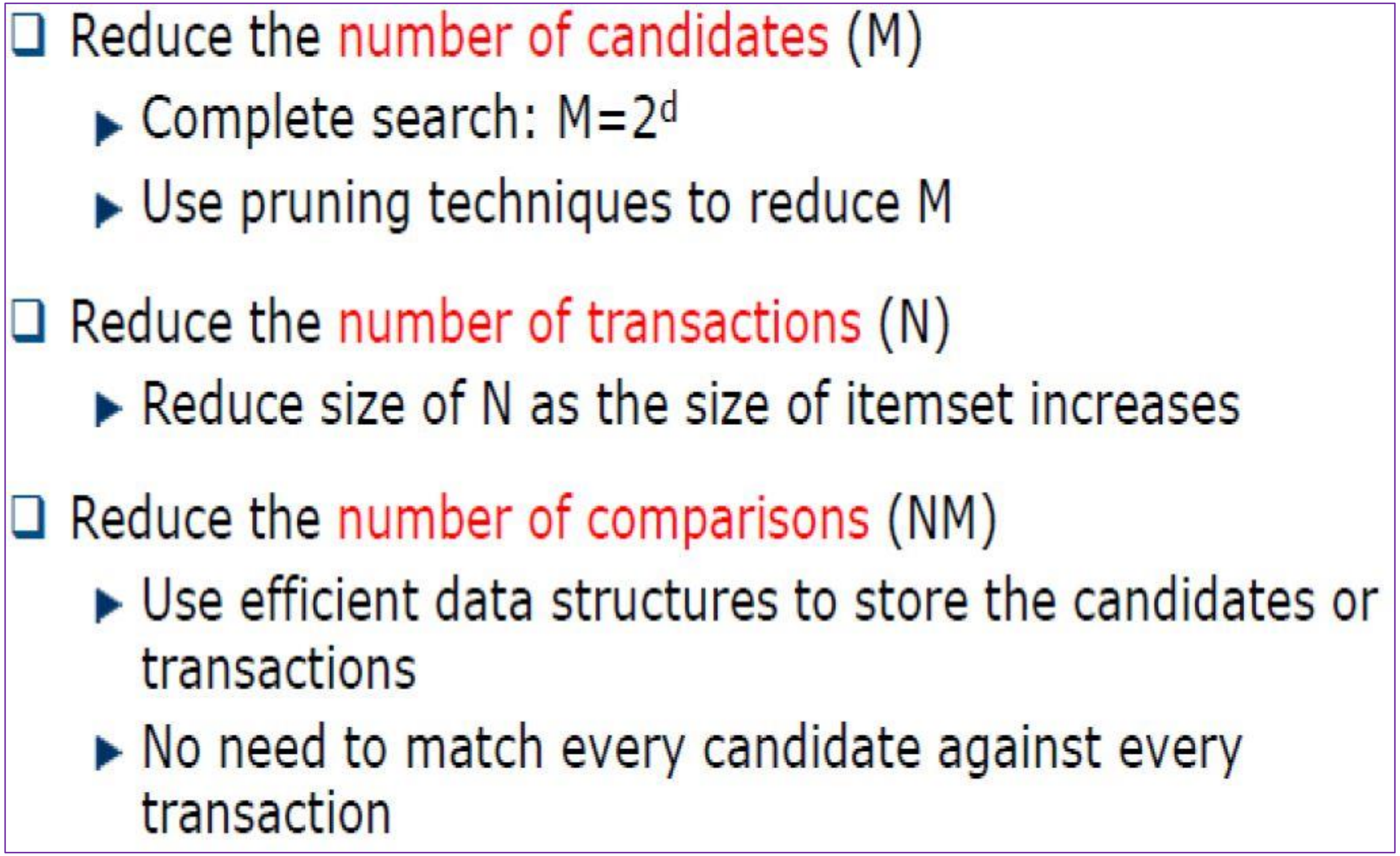
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$

Frequent Itemset Generation Strategies



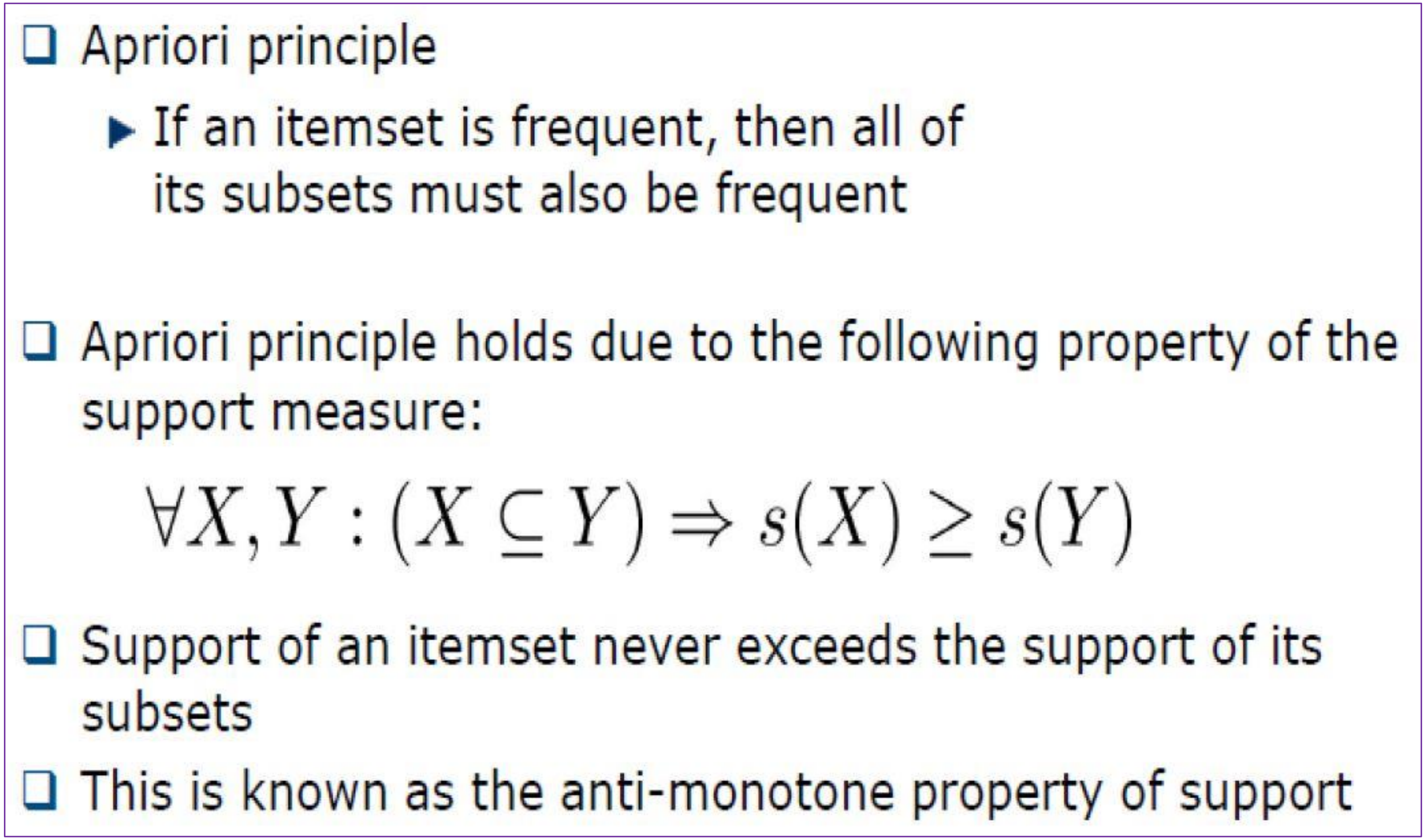
- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Apriori Algorithm

Apriori Algorithm

- This algorithm uses frequent datasets to generate association rules.
- This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently.
- It is mainly used for market basket analysis and helps to understand the products that can be bought together.
- It can also be used in the healthcare field to find drug reactions for patients.

Reducing the Number of Candidates

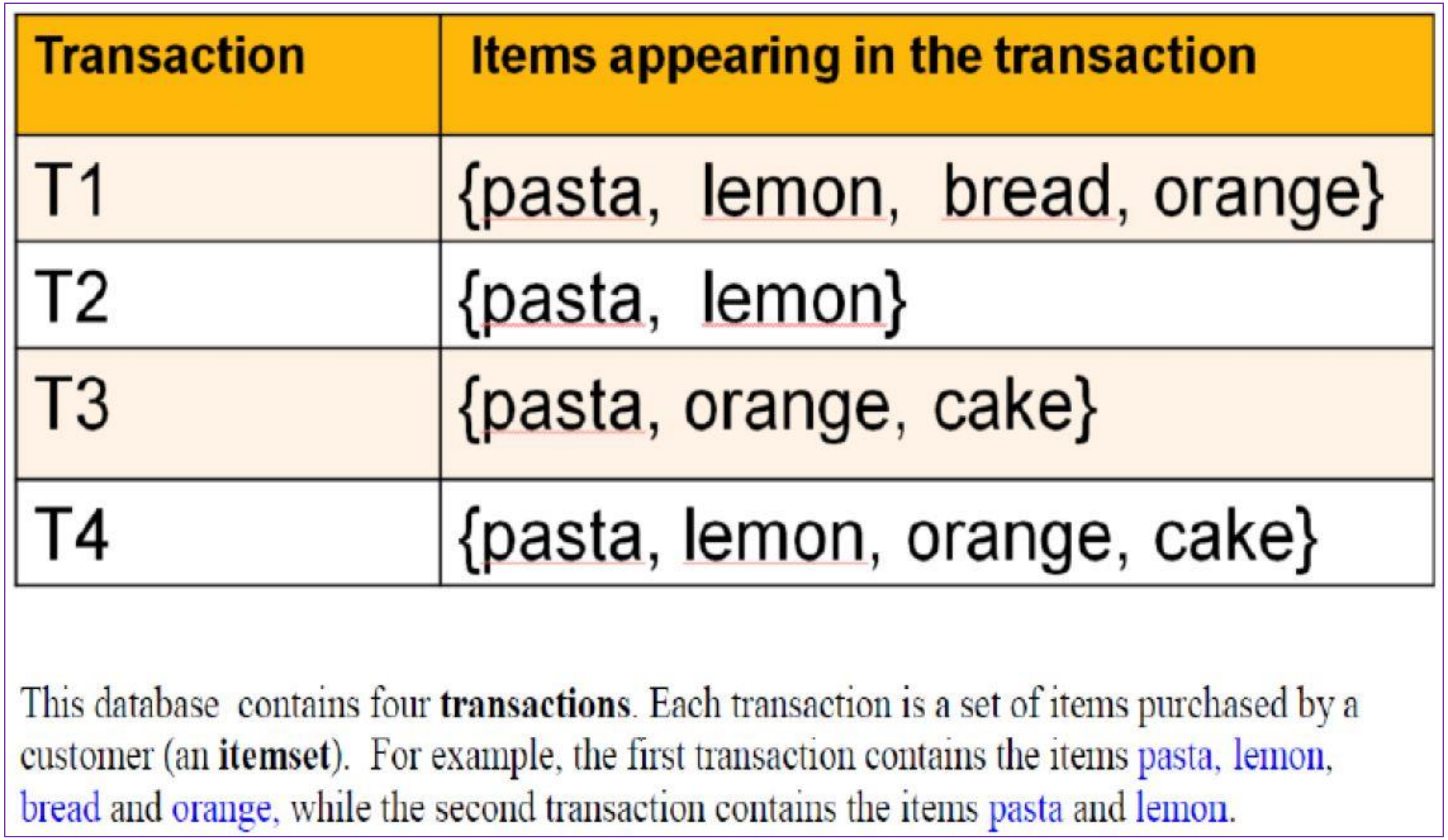


- Apriori principle
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Support of an itemset never exceeds the support of its subsets
- This is known as the anti-monotone property of support

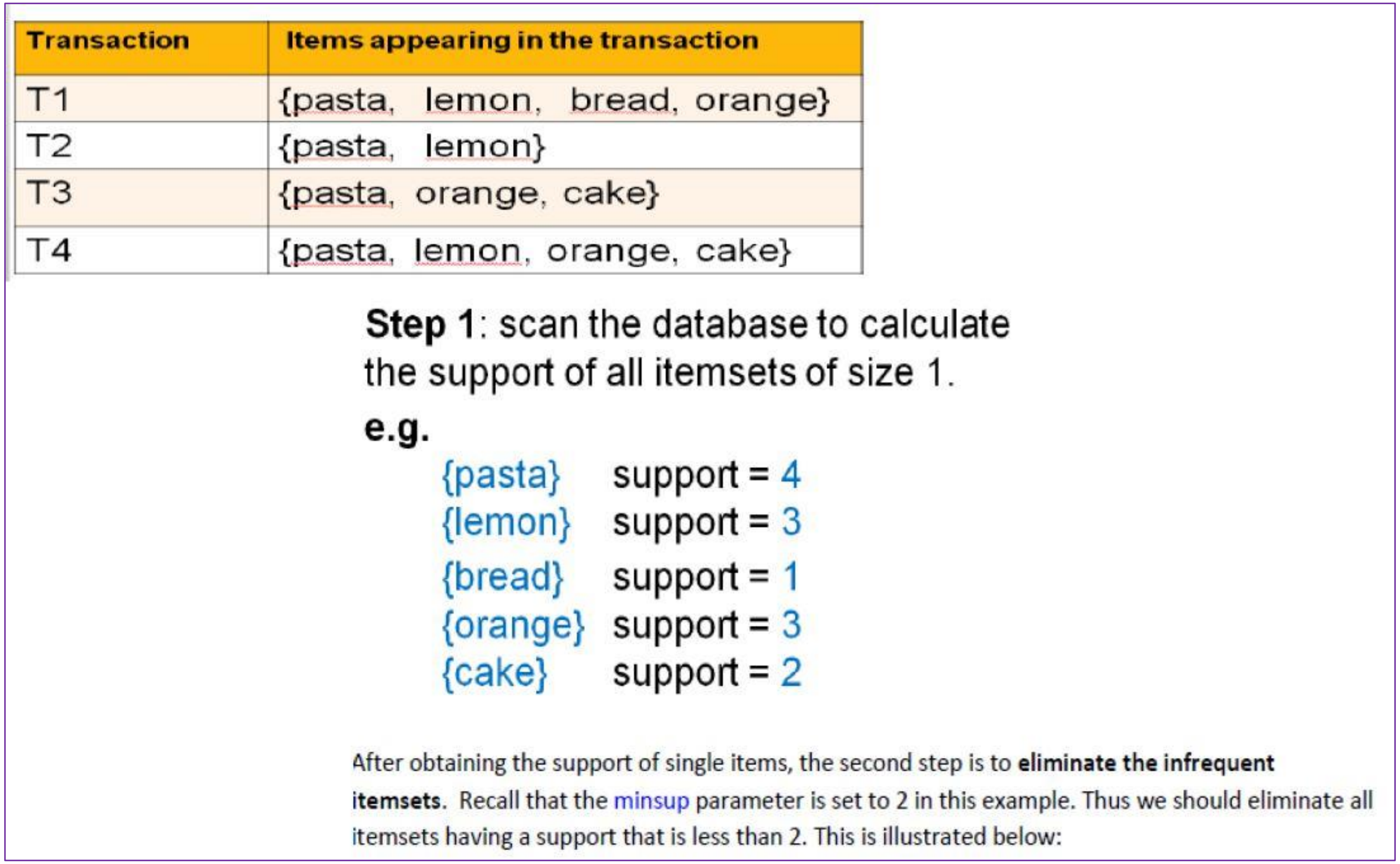
Example: Step by Step



Transaction	Items appearing in the transaction
T1	{pasta, lemon, bread, orange}
T2	{pasta, lemon}
T3	{pasta, orange, cake}
T4	{pasta, lemon, orange, cake}

This database contains four **transactions**. Each transaction is a set of items purchased by a customer (an **itemset**). For example, the first transaction contains the items pasta, lemon, bread and orange, while the second transaction contains the items pasta and lemon.

Step by Step



Transaction	Items appearing in the transaction
T1	{pasta, lemon, bread, orange}
T2	{pasta, lemon}
T3	{pasta, orange, cake}
T4	{pasta, lemon, orange, cake}

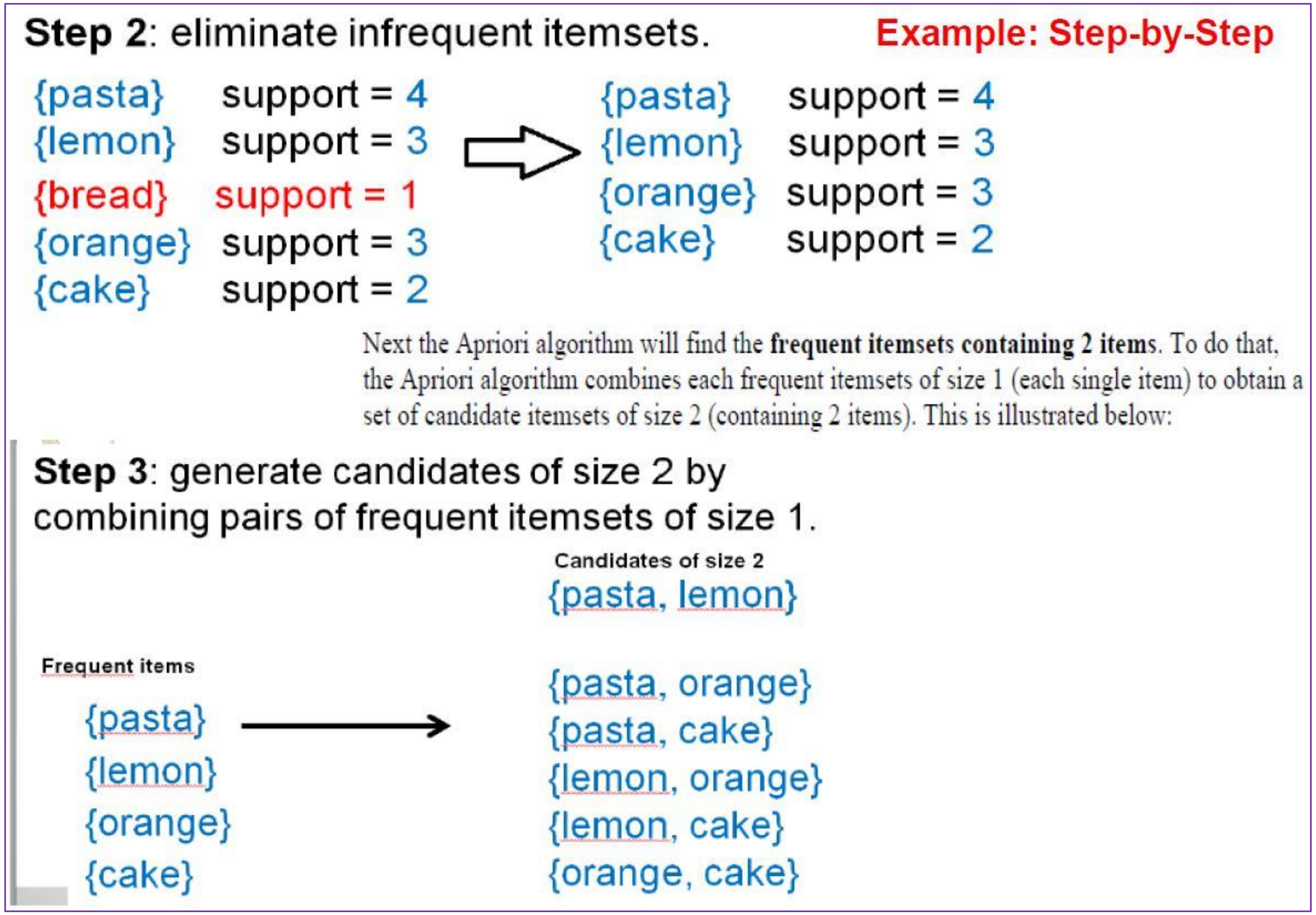
Step 1: scan the database to calculate the support of all itemsets of size 1.

e.g.

{pasta}	support = 4
{lemon}	support = 3
{bread}	support = 1
{orange}	support = 3
{cake}	support = 2

After obtaining the support of single items, the second step is to **eliminate the infrequent itemsets**. Recall that the minsup parameter is set to 2 in this example. Thus we should eliminate all itemsets having a support that is less than 2. This is illustrated below:

Step by Step



Step 2: eliminate infrequent itemsets.

Example: Step-by-Step

{pasta}	support = 4
{lemon}	support = 3
{bread}	support = 1
{orange}	support = 3
{cake}	support = 2

$\Rightarrow$

{pasta}	support = 4
{lemon}	support = 3
{orange}	support = 3
{cake}	support = 2

Next the Apriori algorithm will find the **frequent itemsets containing 2 items**. To do that, the Apriori algorithm combines each frequent itemsets of size 1 (each single item) to obtain a set of candidate itemsets of size 2 (containing 2 items). This is illustrated below:

Step 3: generate candidates of size 2 by combining pairs of frequent itemsets of size 1.

Frequent items

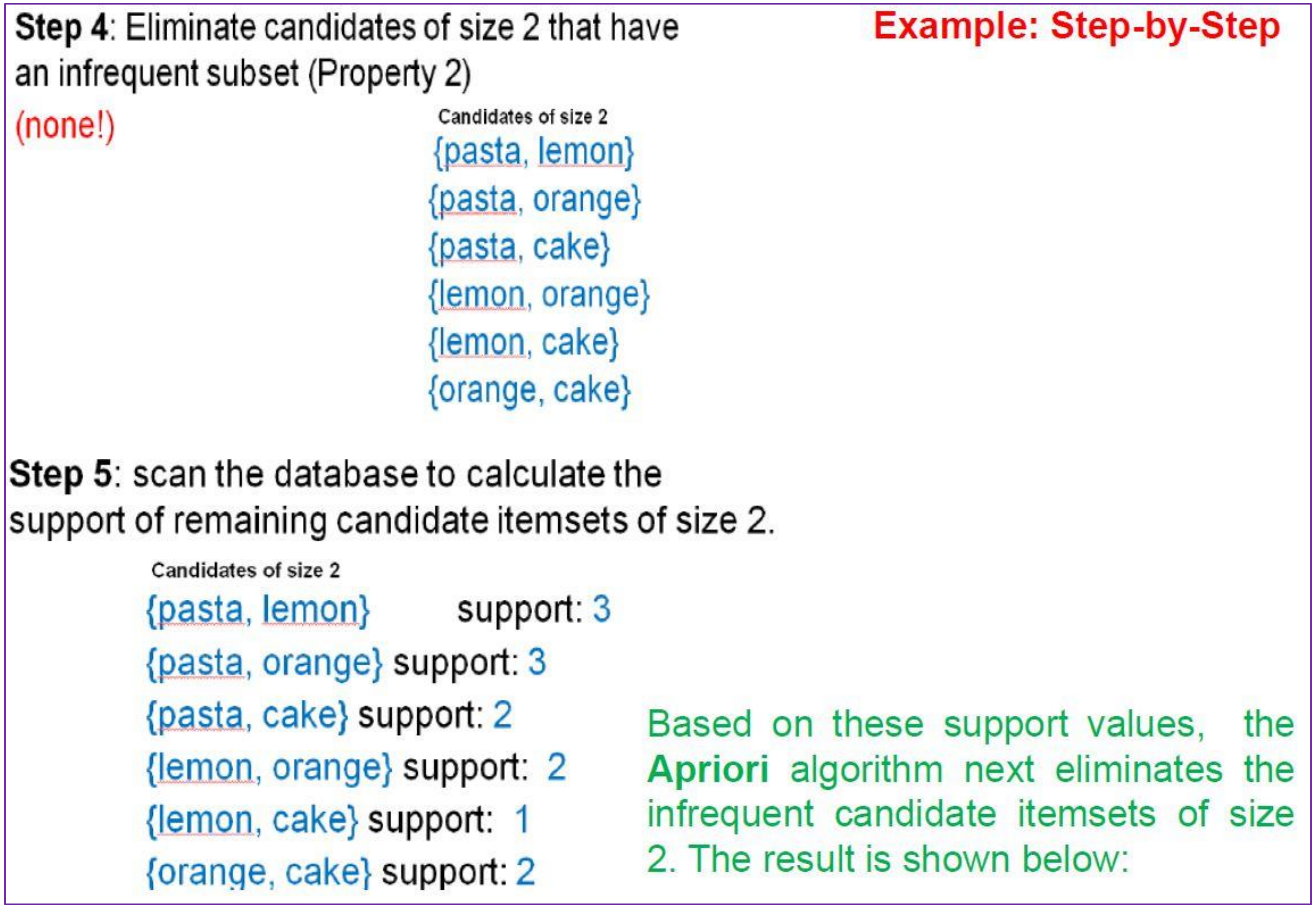
{pasta}
{lemon}
{orange}
{cake}

$\longrightarrow$

Candidates of size 2

{pasta, lemon}
{pasta, orange}
{pasta, cake}
{lemon, orange}
{lemon, cake}
{orange, cake}

Step by Step



Example: Step-by-Step

Step 4: Eliminate candidates of size 2 that have an infrequent subset (Property 2)

(none!)

Candidates of size 2

{pasta, lemon}
{pasta, orange}
{pasta, cake}
{lemon, orange}
{lemon, cake}
{orange, cake}

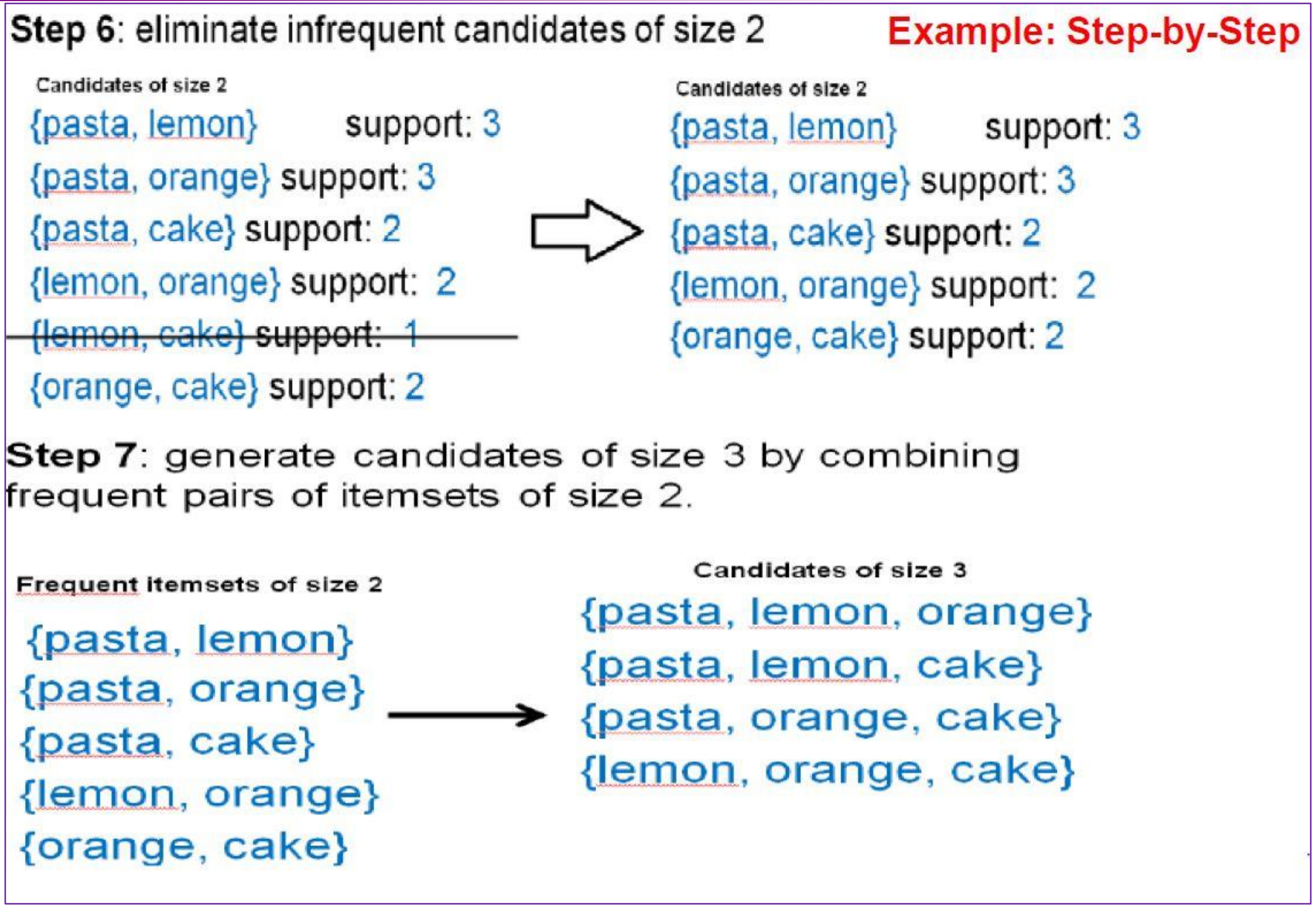
Step 5: scan the database to calculate the support of remaining candidate itemsets of size 2.

Candidates of size 2

{pasta, lemon}	support: 3
{pasta, orange}	support: 3
{pasta, cake}	support: 2
{lemon, orange}	support: 2
{lemon, cake}	support: 1
{orange, cake}	support: 2

Based on these support values, the **Apriori** algorithm next eliminates the infrequent candidate itemsets of size 2. The result is shown below:

Step by Step



Step 6: eliminate infrequent candidates of size 2

Example: Step-by-Step

Candidates of size 2

{pasta, lemon} support: 3
{pasta, orange} support: 3
{pasta, cake} support: 2
{lemon, orange} support: 2
~~{lemon, cake} support: 1~~
{orange, cake} support: 2

Candidates of size 2

{pasta, lemon} support: 3
{pasta, orange} support: 3
{pasta, cake} support: 2
{lemon, orange} support: 2
{orange, cake} support: 2

Step 7: generate candidates of size 3 by combining frequent pairs of itemsets of size 2.

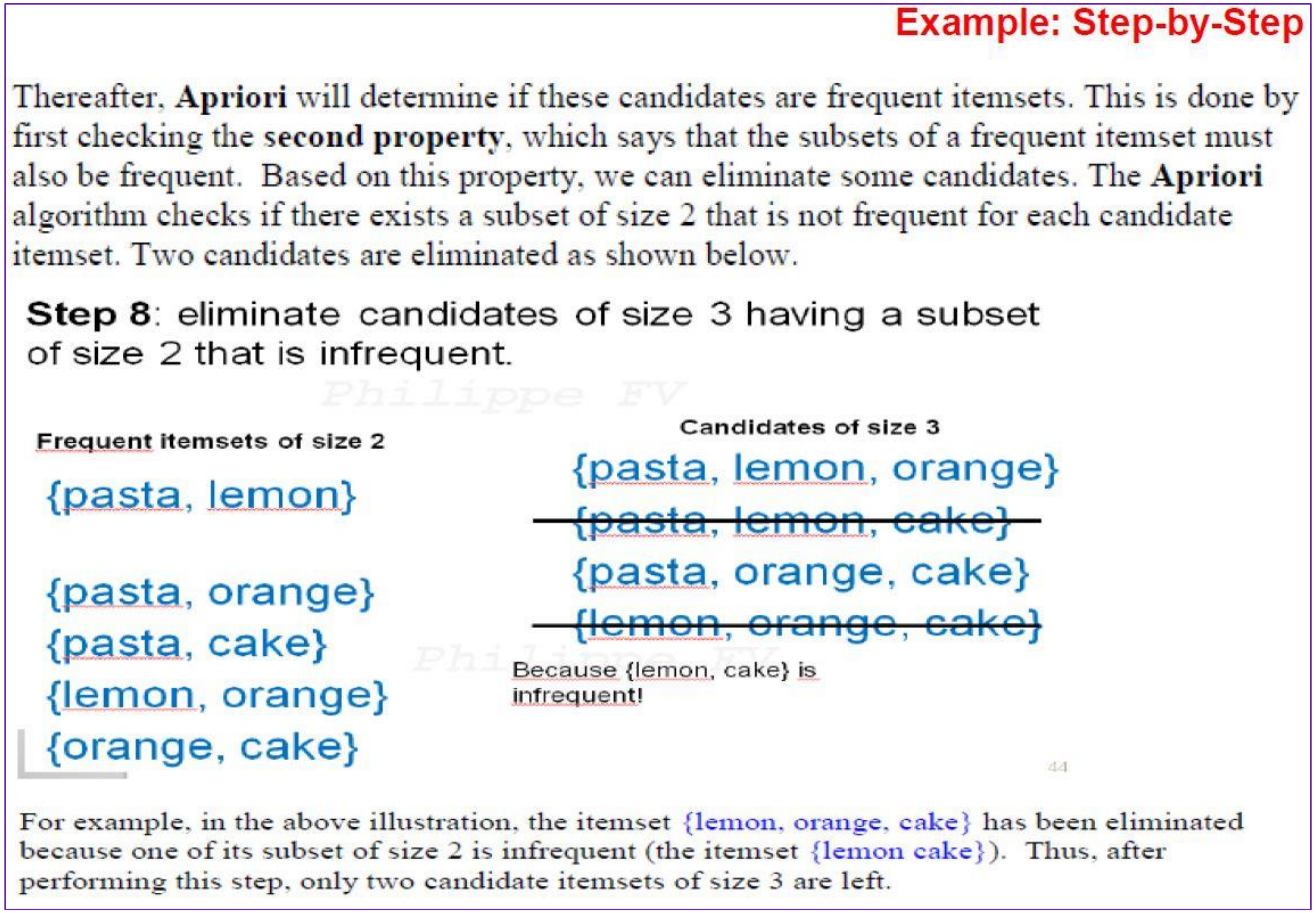
Frequent itemsets of size 2

{pasta, lemon}
{pasta, orange}
{pasta, cake}
{lemon, orange}
{orange, cake}

Candidates of size 3

{pasta, lemon, orange}
{pasta, lemon, cake}
{pasta, orange, cake}
{lemon, orange, cake}

Step by Step



Example: Step-by-Step

Thereafter, **Apriori** will determine if these candidates are frequent itemsets. This is done by first checking the **second property**, which says that the subsets of a frequent itemset must also be frequent. Based on this property, we can eliminate some candidates. The **Apriori** algorithm checks if there exists a subset of size 2 that is not frequent for each candidate itemset. Two candidates are eliminated as shown below.

Step 8: eliminate candidates of size 3 having a subset of size 2 that is infrequent.

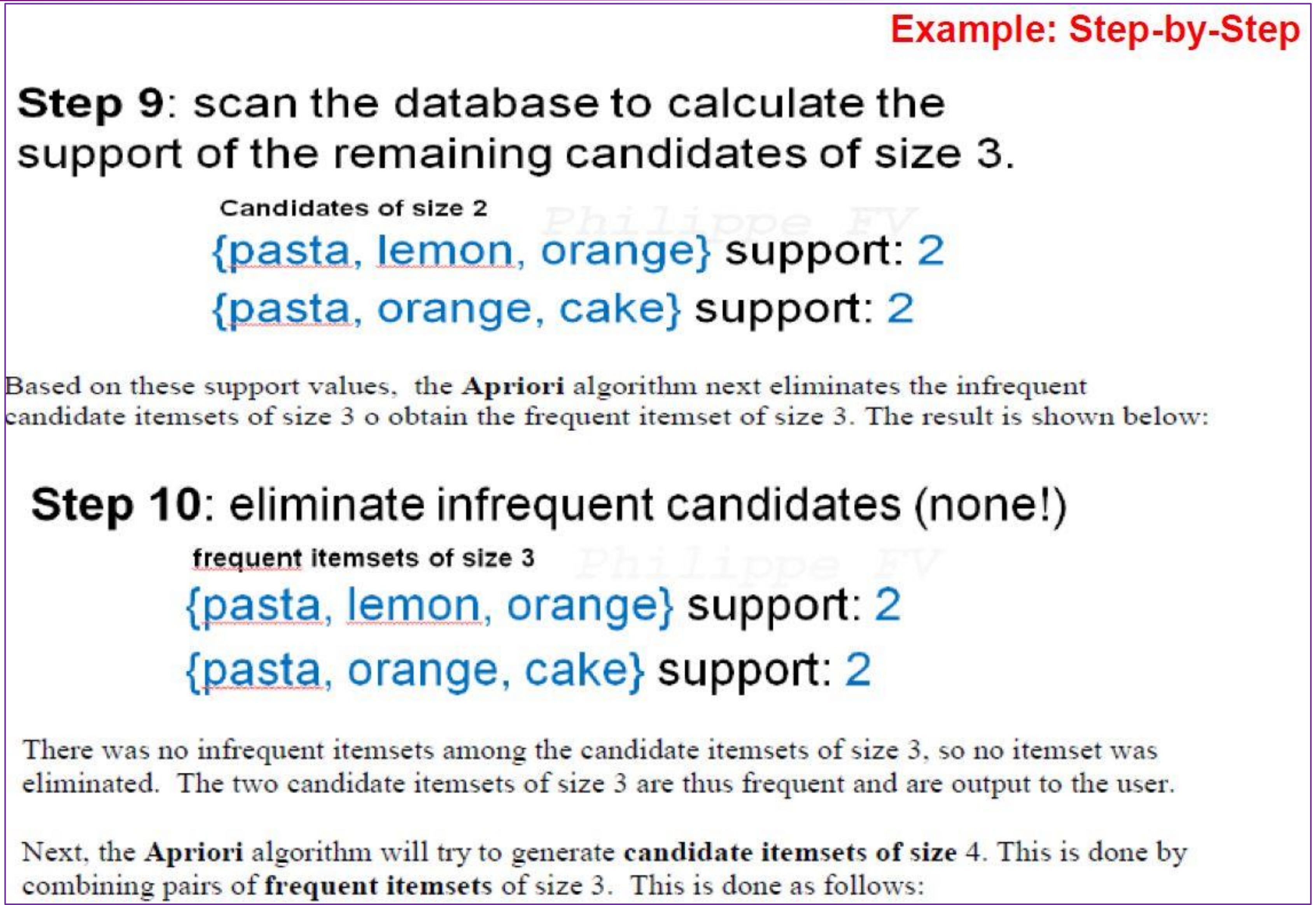
Philippe FV

Frequent itemsets of size 2	Candidates of size 3
{pasta, lemon}	{pasta, lemon, orange}
	~~{pasta, lemon, cake}~~
{pasta, orange}	{pasta, orange, cake}
{pasta, cake}	~~{lemon, orange, cake}~~
{lemon, orange}	Because {lemon, cake} is infrequent!
{orange, cake}	

44

For example, in the above illustration, the itemset {lemon, orange, cake} has been eliminated because one of its subset of size 2 is infrequent (the itemset {lemon cake}). Thus, after performing this step, only two candidate itemsets of size 3 are left.

Step by Step



Example: Step-by-Step

Step 9: scan the database to calculate the support of the remaining candidates of size 3.

Candidates of size 2

{pasta, lemon, orange} support: 2

{pasta, orange, cake} support: 2

Based on these support values, the **Apriori** algorithm next eliminates the infrequent candidate itemsets of size 3 o obtain the frequent itemset of size 3. The result is shown below:

Step 10: eliminate infrequent candidates (none!)

frequent itemsets of size 3

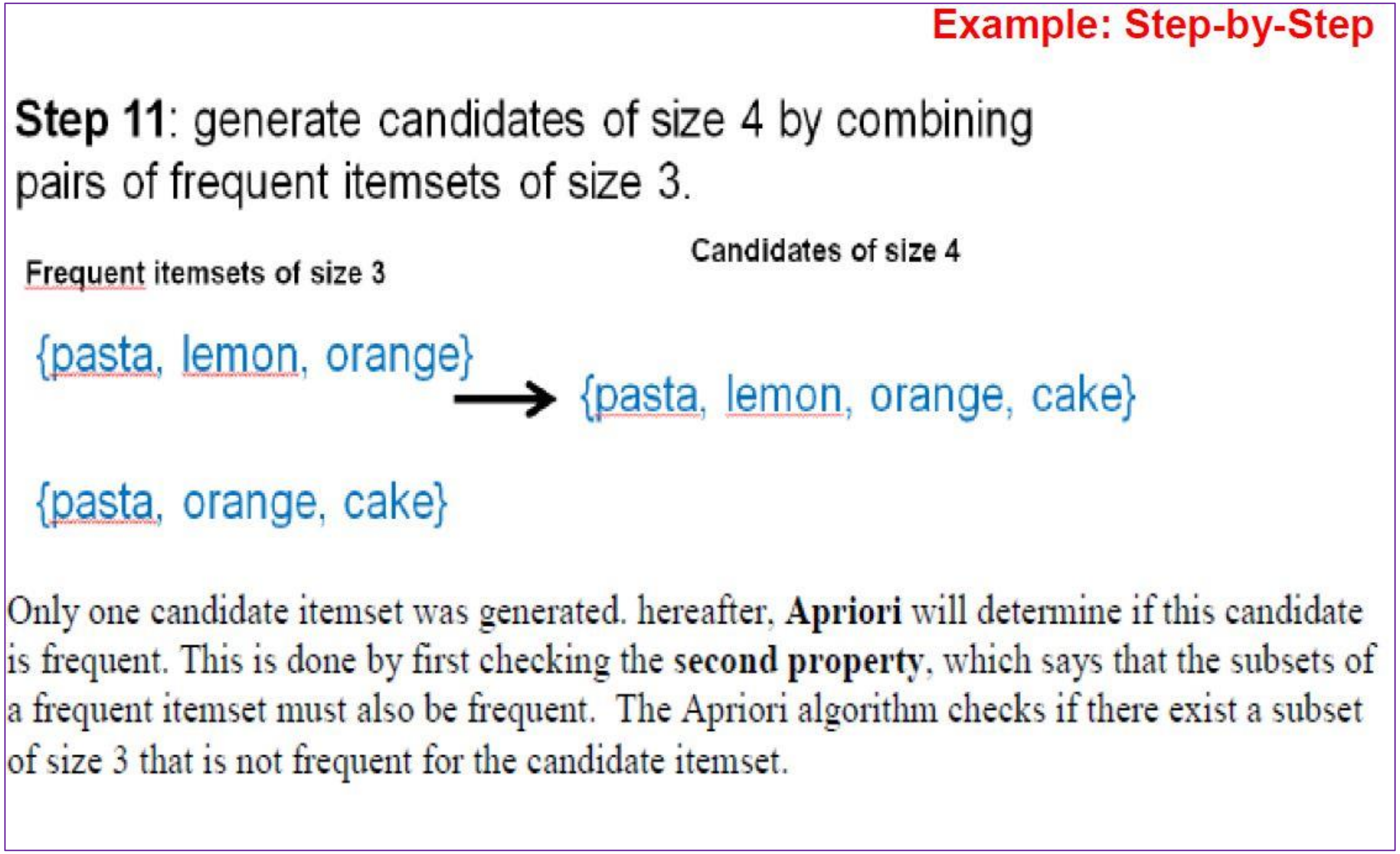
{pasta, lemon, orange} support: 2

{pasta, orange, cake} support: 2

There was no infrequent itemsets among the candidate itemsets of size 3, so no itemset was eliminated. The two candidate itemsets of size 3 are thus frequent and are output to the user.

Next, the **Apriori** algorithm will try to generate **candidate itemsets of size** 4. This is done by combining pairs of **frequent itemsets** of size 3. This is done as follows:

Step by Step



Example: Step-by-Step

Step 11: generate candidates of size 4 by combining pairs of frequent itemsets of size 3.

Frequent itemsets of size 3

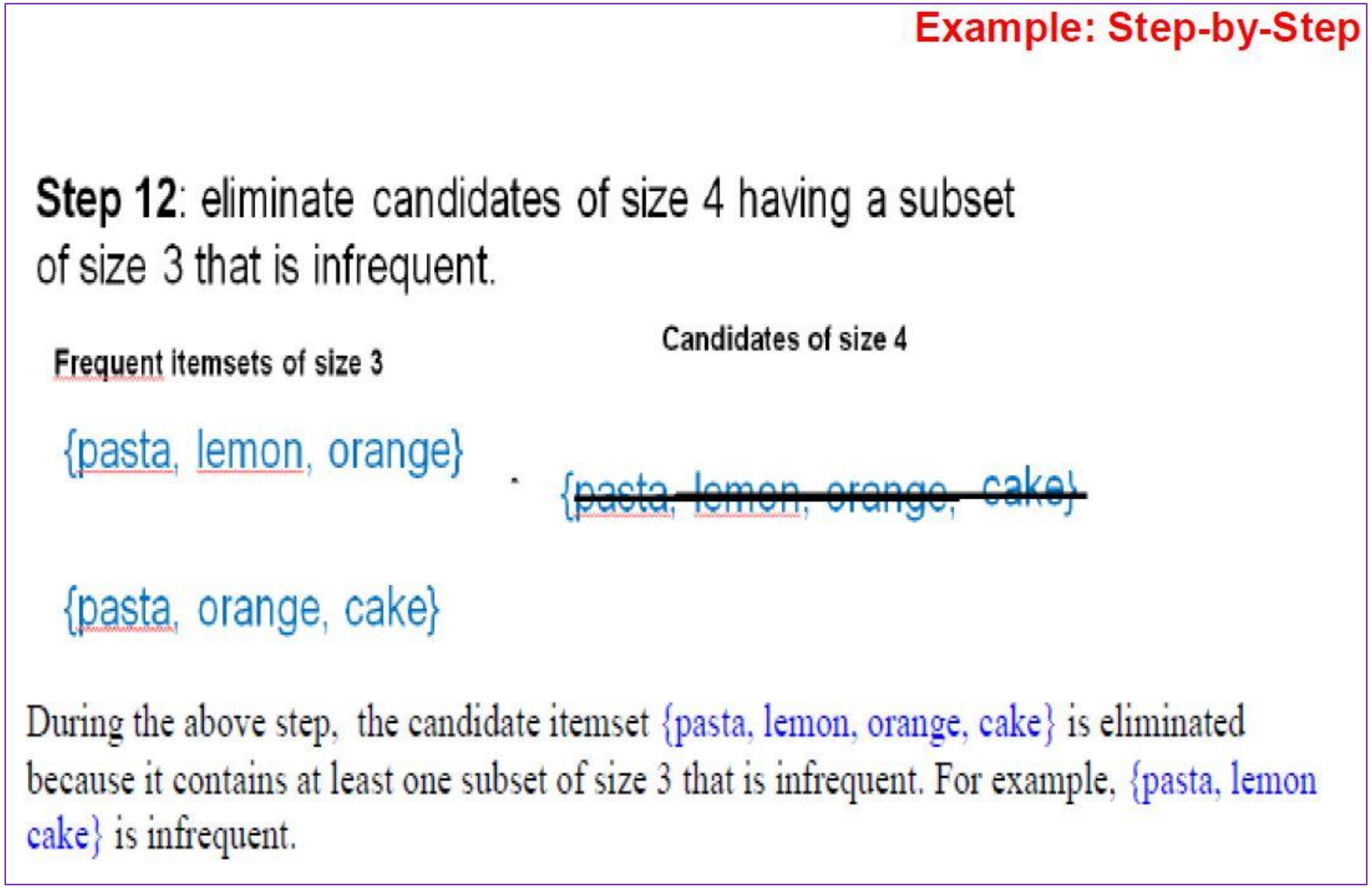
Candidates of size 4

{pasta, lemon, orange} $\rightarrow$ {pasta, lemon, orange, cake}

{pasta, orange, cake}

Only one candidate itemset was generated. hereafter, **Apriori** will determine if this candidate is frequent. This is done by first checking the **second property**, which says that the subsets of a frequent itemset must also be frequent. The Apriori algorithm checks if there exist a subset of size 3 that is not frequent for the candidate itemset.

Step by Step



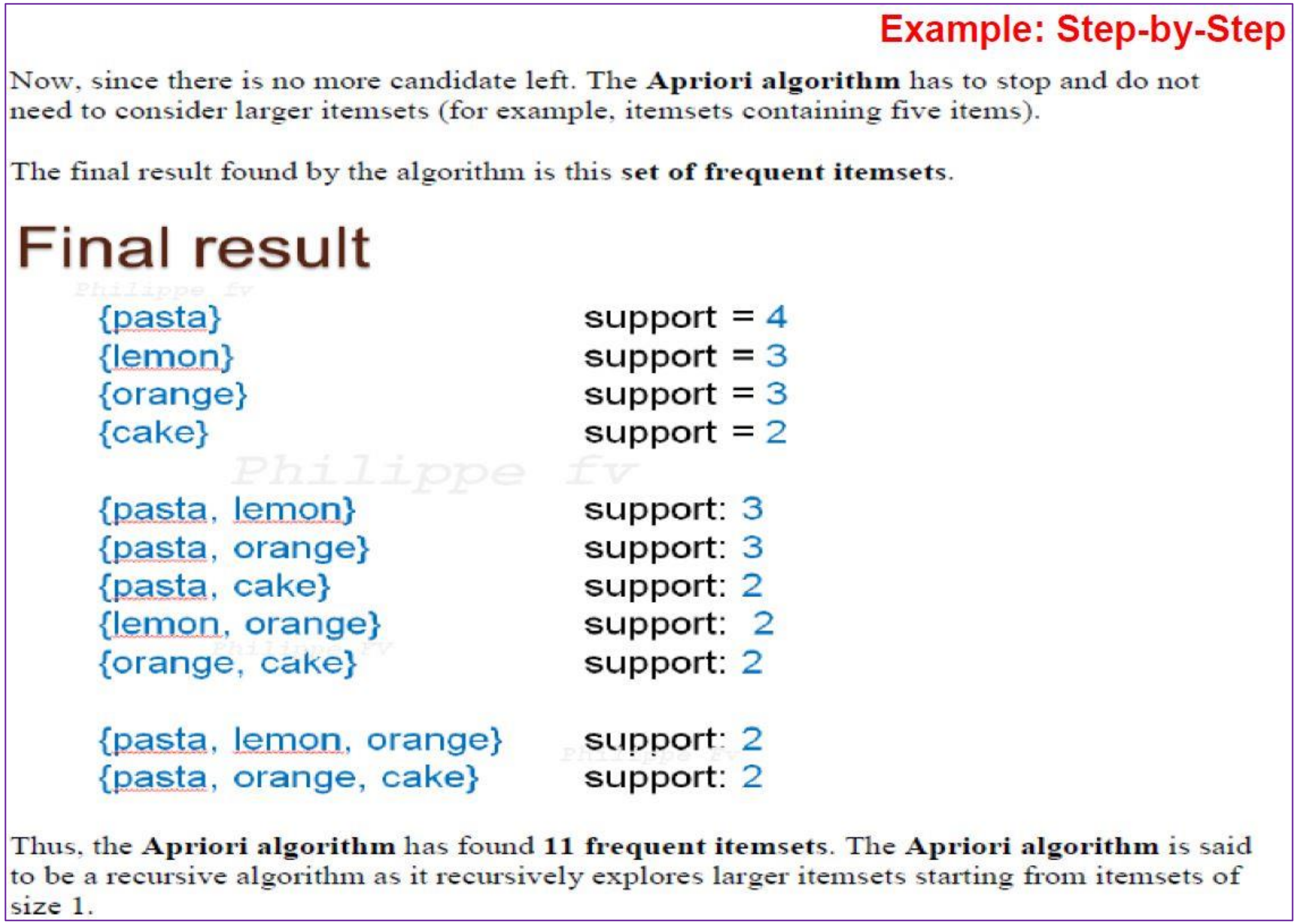
Example: Step-by-Step

Step 12: eliminate candidates of size 4 having a subset of size 3 that is infrequent.

Frequent itemsets of size 3	Candidates of size 4
{pasta, lemon, orange}	~~{pasta, lemon, orange, cake}~~
{pasta, orange, cake}	

During the above step, the candidate itemset {pasta, lemon, orange, cake} is eliminated because it contains at least one subset of size 3 that is infrequent. For example, {pasta, lemon cake} is infrequent.

Step by Step



Example: Step-by-Step

Now, since there is no more candidate left. The **Apriori algorithm** has to stop and do not need to consider larger itemsets (for example, itemsets containing five items).

The final result found by the algorithm is this **set of frequent itemsets**.

Final result

{pasta}	support = 4
{lemon}	support = 3
{orange}	support = 3
{cake}	support = 2
{pasta, lemon}	support: 3
{pasta, orange}	support: 3
{pasta, cake}	support: 2
{lemon, orange}	support: 2
{orange, cake}	support: 2
{pasta, lemon, orange}	support: 2
{pasta, orange, cake}	support: 2

Thus, the **Apriori algorithm** has found **11 frequent itemsets**. The **Apriori algorithm** is said to be a recursive algorithm as it recursively explores larger itemsets starting from itemsets of size 1.

Questions