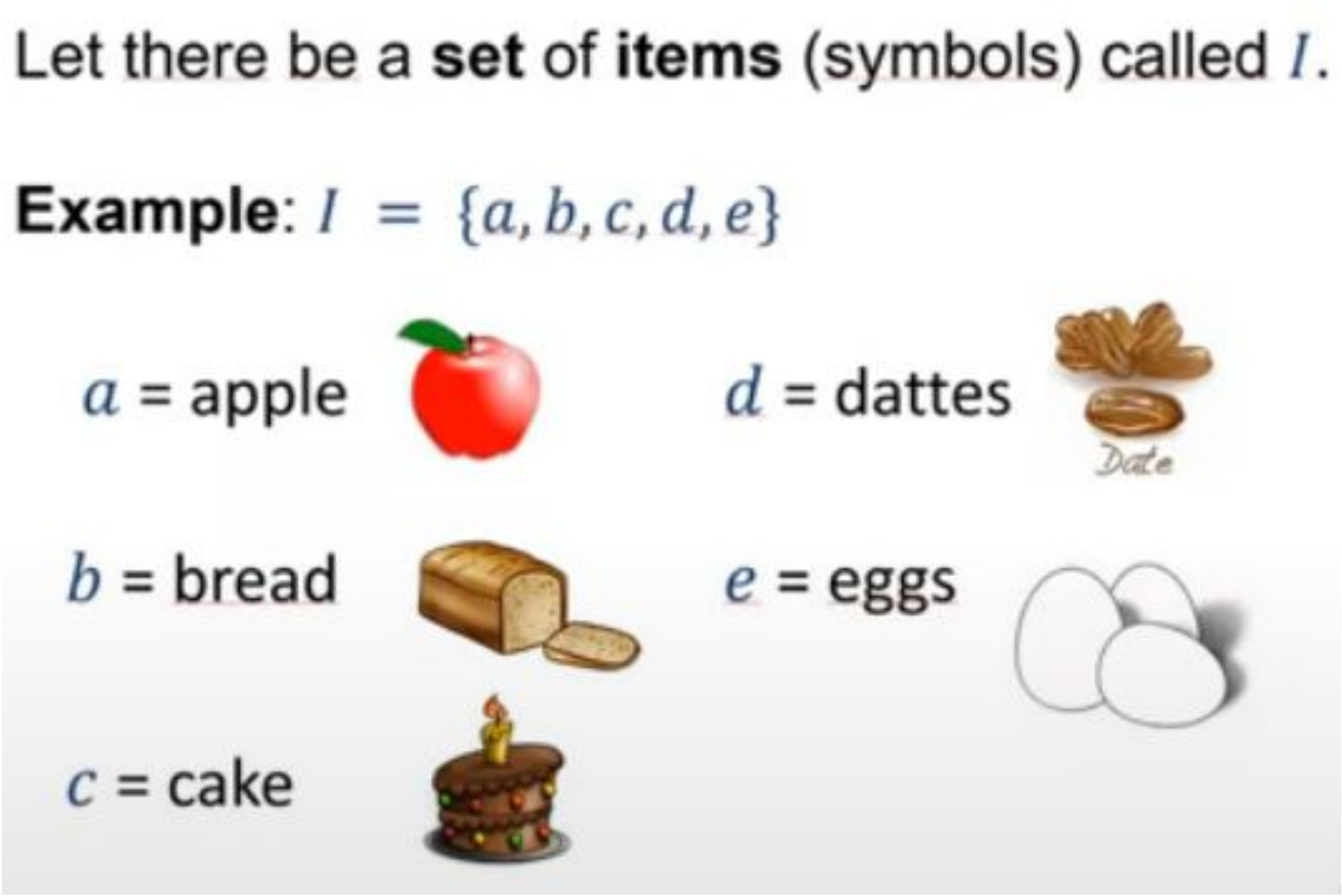
Sequence Pattern Mining (SPM)

What is Sequential Pattern Mining?

- **Sequence:** A sequence is a list of ordered items.
- Sequential pattern mining is the mining of frequently appearing series events or subsequences as patterns. An instance of a sequential pattern is users who purchase a Canon digital camera are to purchase an HP color printer within a month.
- For retail information, sequential patterns are beneficial for shelf placement and promotions. This industry, and telecommunications and different businesses, can also use sequential patterns for targeted marketing, user retention, and several tasks.
- There are several areas in which sequential patterns can be used such as Web access pattern analysis, weather prediction, production processes, and web intrusion detection.



Let there be a **set** of **items** (symbols) called I.

Example: $I = \{a, b, c, d, e\}$

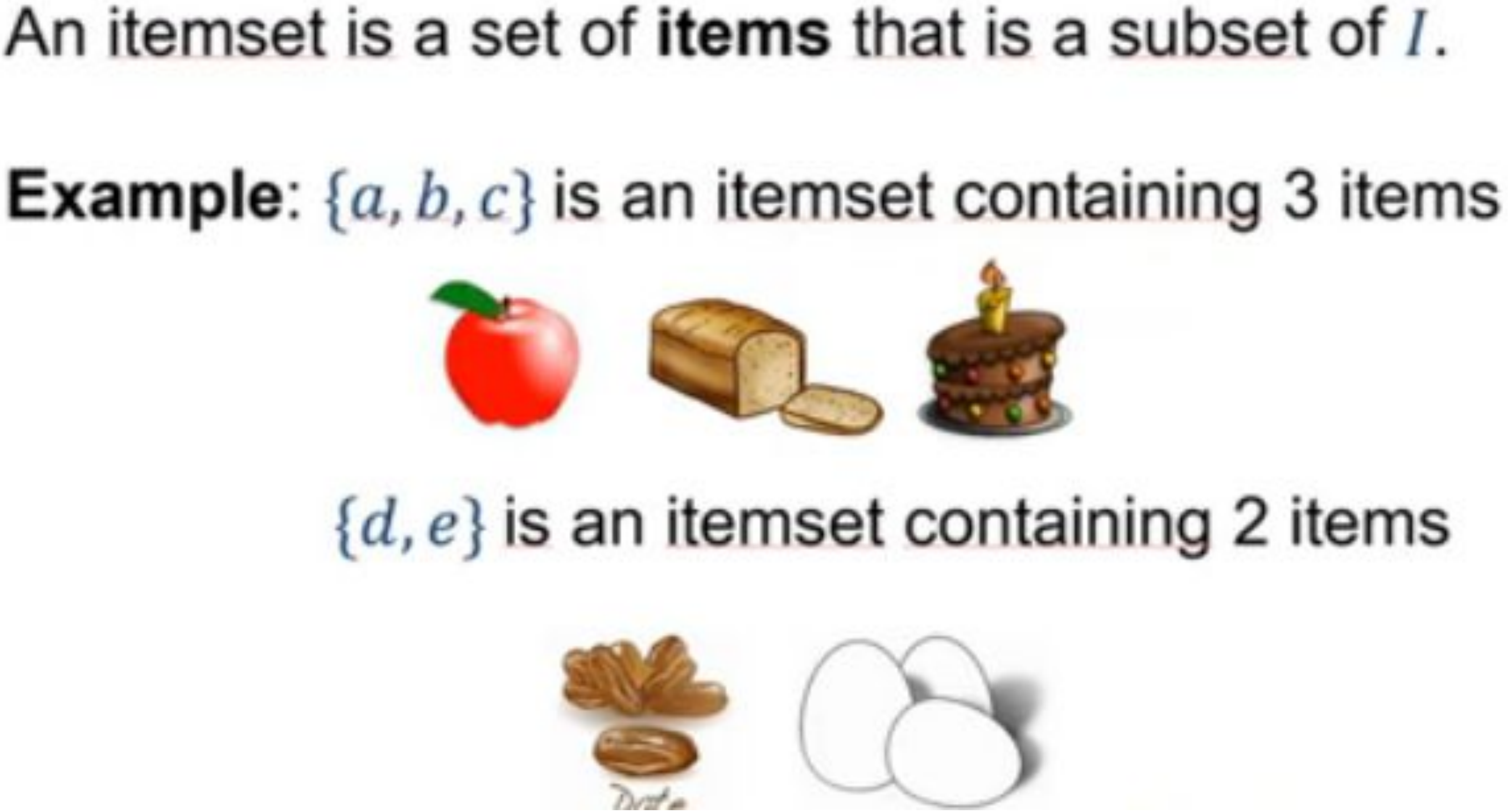
a = apple

b = bread

c = cake

d = dattes

e = eggs



An itemset is a set of **items** that is a subset of I.

Example: $\{a, b, c\}$ is an itemset containing 3 items

$\{d, e\}$ is an itemset containing 2 items

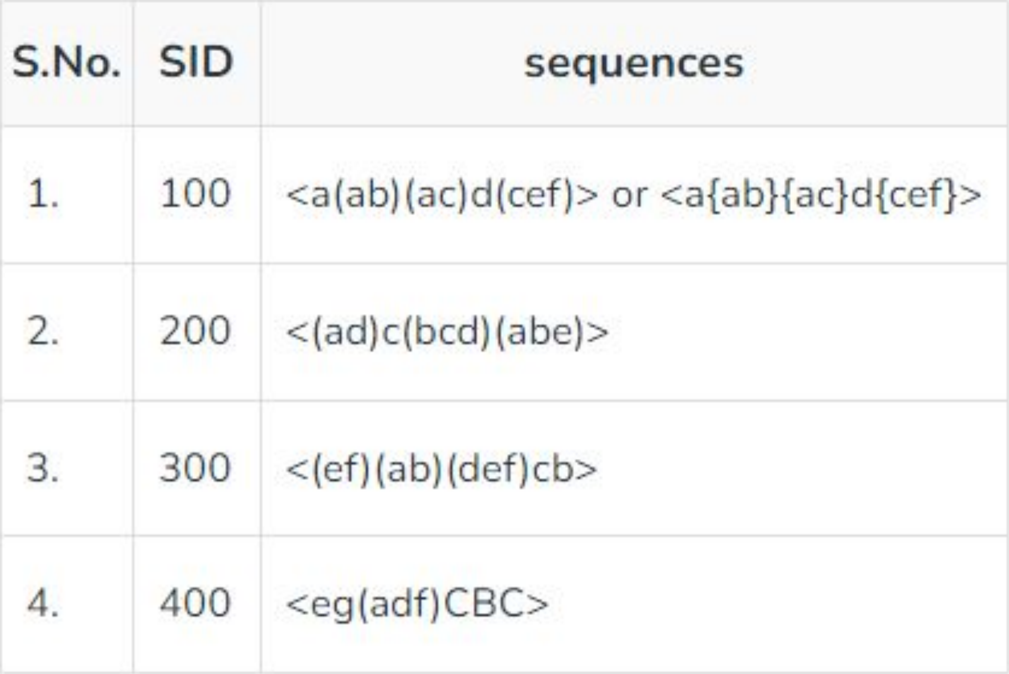
Definition: Sequence

A **discrete sequence** S is a an ordered list of itemsets $S = \langle X_1, X_2, \ldots, X_n \rangle$ where $X_j \subseteq I$ for any $j \in \{1,2..n\}$

Example 1: $\langle \{a, b\}, \{c\} \rangle$ is a sequence containing two itemsets.

It means that a customer purchased $apple$ and $bread$ at the same time and then purchased $cake$.

- **Sequence Database:** A database that consists of ordered elements or events is called a sequence database. Example of a sequence database:



S.No.	SID	sequences
1.	100	<a(ab)(ac)d(cef)> or <a{ab}{ac}d{cef}>
2.	200	<(ad)c(bcd)(abe)>
3.	300	<(ef)(ab)(def)cb>
4.	400	<eg(adf)CBC>

Transaction: The sequence consists of many elements which are called transactions.

<a(ab)(ac)d(cef)> is a sequence whereas (a), (ab), (ac),

(d) and (cef) are the elements of the sequence.

These elements are sometimes referred as transactions.

An element may contain a set of items. Items within an element are unordered and we list them alphabetically.

For example, (cef) is the element and it consists of 3 items c, e and f.

***Since, all three items belong to same element, their order does not matter.** But we prefer to put them in alphabetical order for convenience.*

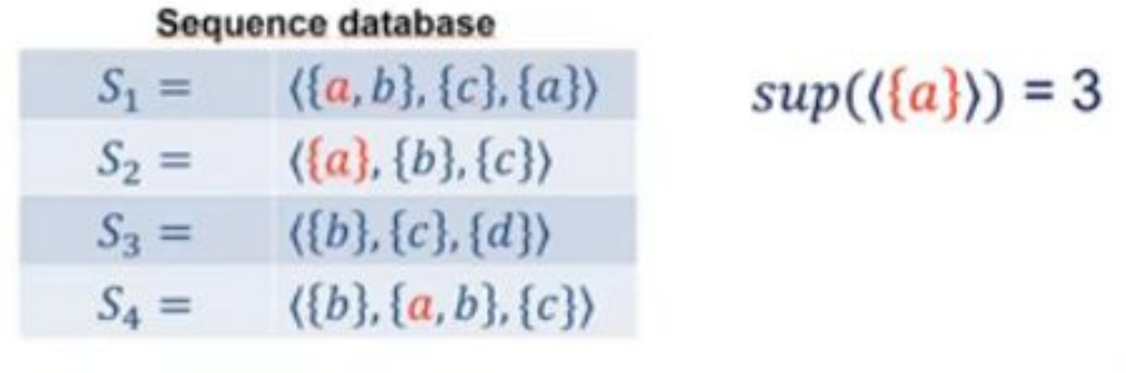
The order of the elements of the sequence matters unlike order of items in same transaction.

k-length Sequence:

The number of items involved in the sequence is denoted by K. A sequence of 2 items is called a 2-len sequence. While finding the 2-length candidate sequence this term comes into use. Example of 2-length sequence is: {ab}, {(ab)}, {bc} and {(bc)}.

- {bc} denotes a 2-length sequence where b and c are two different transactions. This can also be written as {(b)(c)}
- {(bc)} denotes a 2-length sequence where b and c are the items belonging to the same transaction, therefore enclosed in the same parenthesis. This can also be written as {(cb)}, because the order of items in the same transaction does not matter.

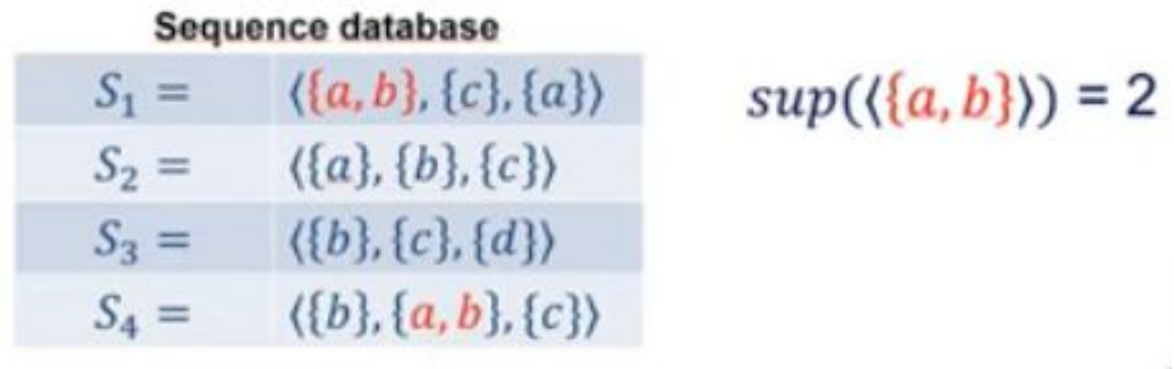
Example 1:



Sequence database

$S_1 =$	$\langle\{a,b\},\{c\},\{a\}\rangle$
$S_2 =$	$\langle\{a\},\{b\},\{c\}\rangle$
$S_3 =$	$\langle\{b\},\{c\},\{d\}\rangle$
$S_4 =$	$\langle\{b\},\{a,b\},\{c\}\rangle$

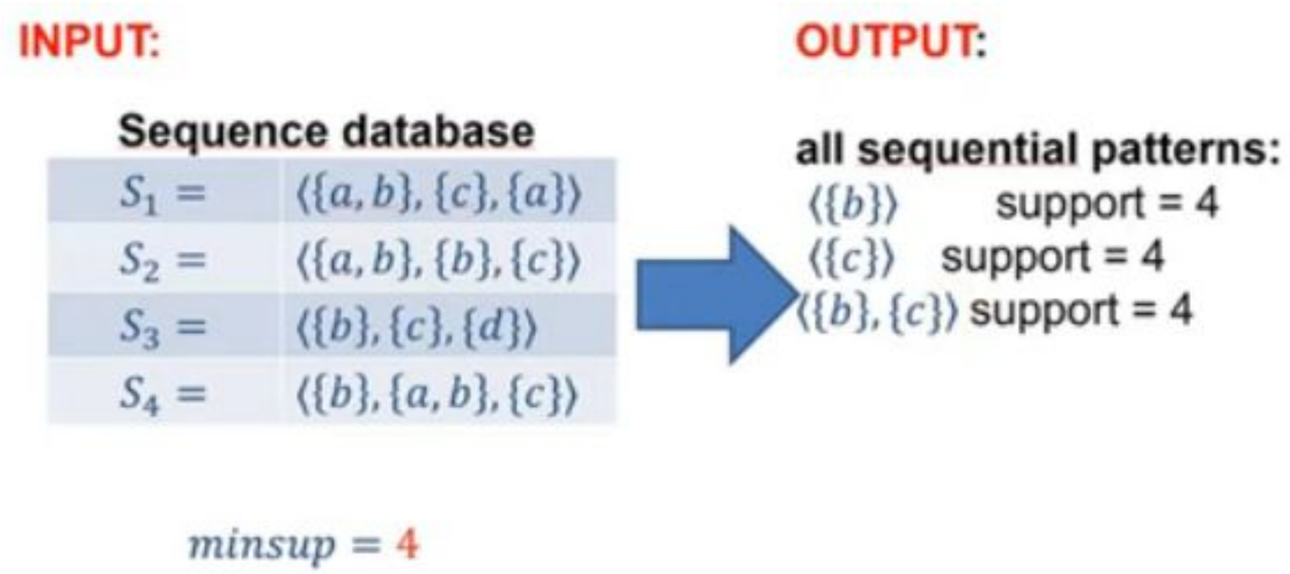
$sup(\langle\{a\}\rangle) = 3$



Sequence database

$S_1 =$	$\langle\{a,b\},\{c\},\{a\}\rangle$
$S_2 =$	$\langle\{a\},\{b\},\{c\}\rangle$
$S_3 =$	$\langle\{b\},\{c\},\{d\}\rangle$
$S_4 =$	$\langle\{b\},\{a,b\},\{c\}\rangle$

$sup(\langle\{a,b\}\rangle) = 2$



INPUT:

Sequence database

$S_1 =$	$\langle\{a,b\},\{c\},\{a\}\rangle$
$S_2 =$	$\langle\{a,b\},\{b\},\{c\}\rangle$
$S_3 =$	$\langle\{b\},\{c\},\{d\}\rangle$
$S_4 =$	$\langle\{b\},\{a,b\},\{c\}\rangle$

$minsup = 4$

OUTPUT:

all sequential patterns:

$\langle\{b\}\rangle$ support = 4

$\langle\{c\}\rangle$ support = 4

$\langle\{b\},\{c\}\rangle$ support = 4

Support in k-length Sequence:

Support means the frequency. The number of occurrences of a given k-length sequence in the sequence database is known as the support. While finding the support the order is taken care.

Illustration:

Suppose we have following sequences in the database. Minimum Support Count = 2.

SID	Sequence
10	<a (**ab**c) (a**c**) d (cf)>
20	<(ad) c (bc) (ae)>
30	<(ef) (**ab**) (df**c**b)>
40	<e g (af) c b c>

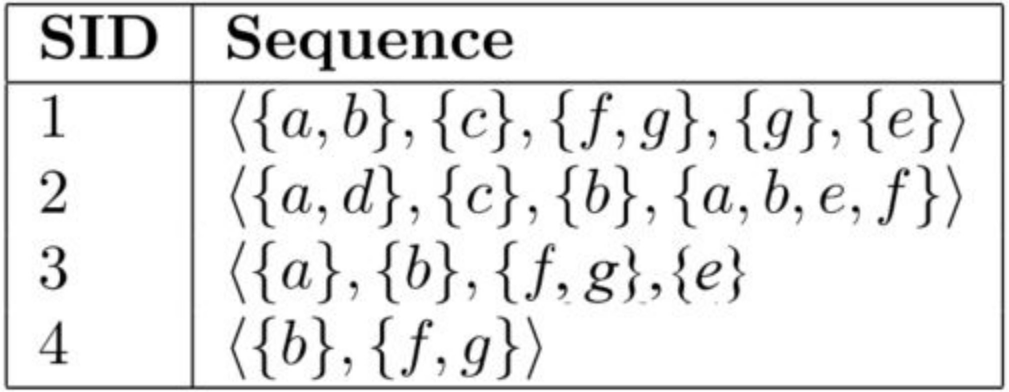
Find if the sub sequence is <(ab) c>

Finding if the sub sequence is <(ab) c>.

The above sequence is the sub sequence, satisfying the condition of min support count = 2.

Note: in SID = 10, we cannot take “c” from the element (abc) because in <(ab) c>, “ab” and “c” are two different elements.

Example:



SID	Sequence
1	$\langle\{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\}\rangle$
2	$\langle\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}\rangle$
3	$\langle\{a\}, \{b\}, \{f, g\},\{e\}$
4	$\langle\{b\}, \{f, g\}\rangle$

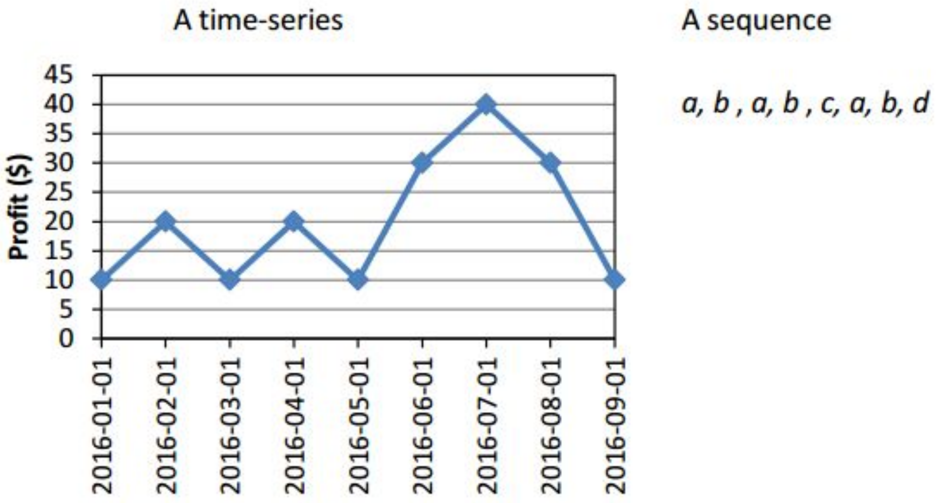
- This database contains four sequences.
- Each **sequence** represents the items purchased by a customer at different times.
- A sequence is an ordered list of itemsets (sets of items bought together).
- For example, in this database, the first sequence (SID 1) indicates that a customer bought some items *a* and *b* together, then purchased an item *c*, then purchased items *f* and *g* together, then purchased an item *g*, and then finally purchased an item *e*.

- **Traditionally, sequential pattern mining** is being used to find subsequences that appear often in a sequence database, i.e. that are common to several sequences. Those subsequences are called the **frequent sequential patterns**. For example, in the context of our example, **sequential pattern mining** can be used to find the sequences of items frequently bought by customers. This can be useful to understand the behavior of customers to take marketing decisions.

- To do **sequential pattern mining**, a user must provide a sequence database and specify a parameter called the **minimum support threshold**.
- This parameter indicates a minimum number of sequences in which a **pattern** must appear to be considered frequent, and be shown to the user.
- For example, if a user sets the minimum support threshold to 2 sequences, the task of **sequential pattern mining** consists of finding all subsequences appearing in at least 2 sequences of the input database.
- In the example database, many subsequences met this requirement. Some of these **sequential** patterns are shown in the table below, where the number of sequences containing each **pattern** (called the *support*) is indicated in the right column of the table**.**

Can sequential pattern mining be applied to time series?

- Besides sequences, **sequential pattern mining** can also be applied to **time series** (e.g. stock data), when discretization is performed as a pre-processing step. For example, the figure below shows a **time series** (an ordered list of numbers) on the left. On the right, a **sequence** (a sequence of symbols) is shown representing the same data, after applying a transformation. Various transformations can be done to transform a time series to a sequence such as the popular SAX transformation (The SAX is **a capital-weighted index that compares the market capitalization of a selected set of shares with** the market capitalization of the same set of shares). After performing the transformation, any **sequential pattern mining** algorithm can be applied.



A time-series (left) and a sequence (right)

Generalized Sequence Pattern Mining (GSP)

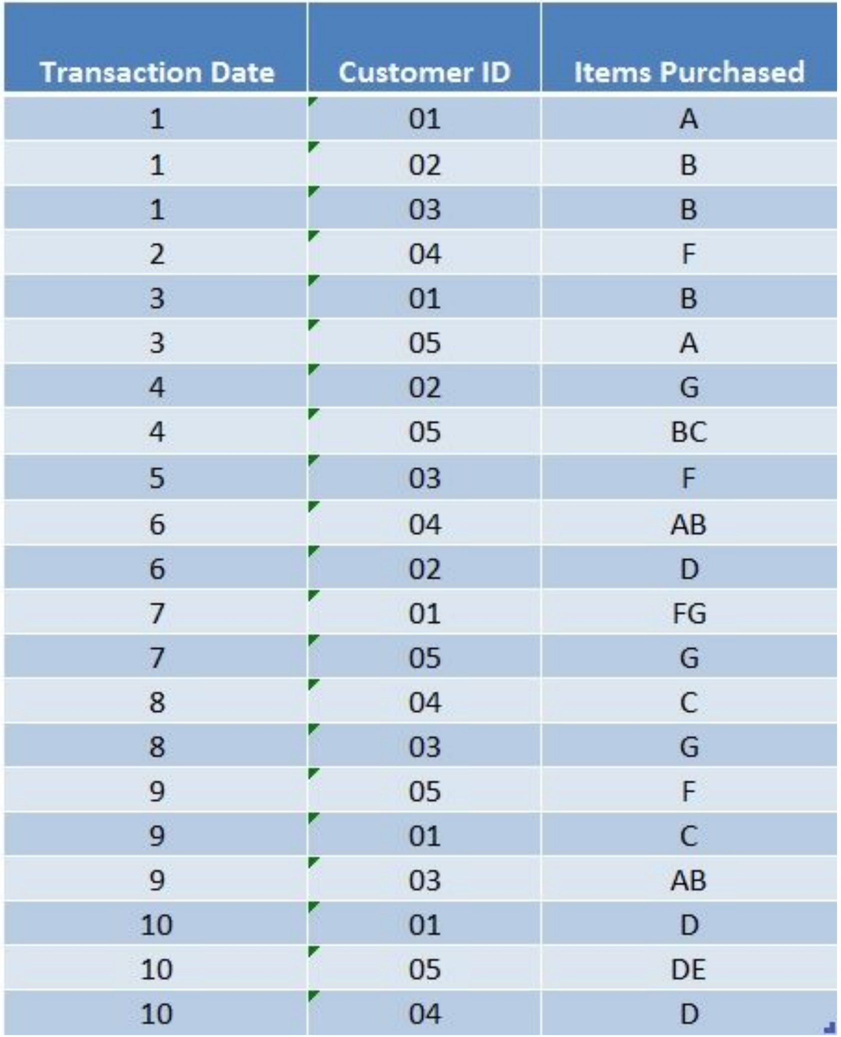
GSP: Horizontal Based Data

- Sequential pattern mining, also known as GSP (Generalized Sequential Pattern) mining, is a technique used to identify patterns in sequential data. The goal of GSP mining is to discover patterns in data that occur over time, such as customer buying habits, website navigation patterns, or sensor data.

Applications of GSP:

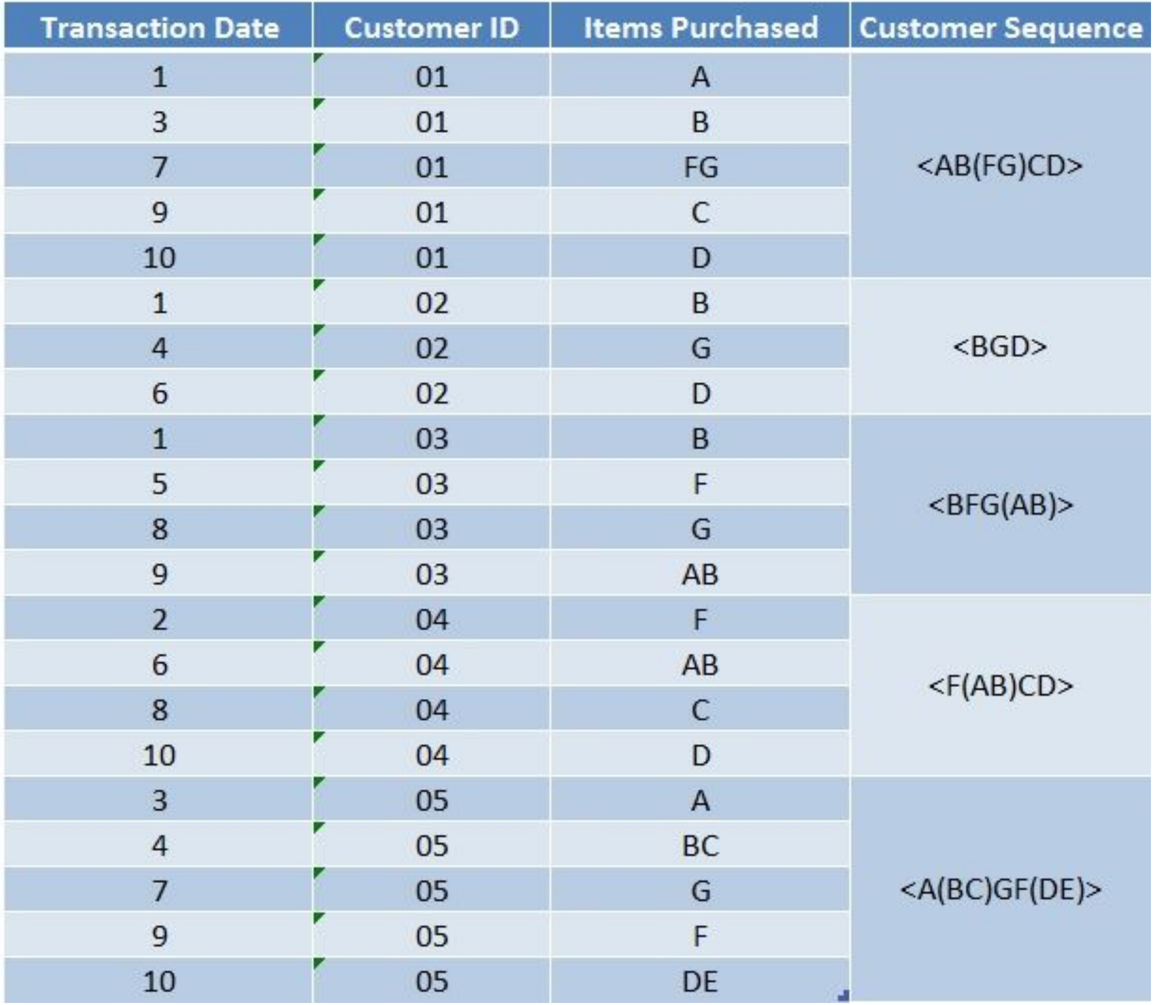
- **Fraud detection:** GSP mining can be used to identify patterns of behavior that are indicative of fraud, such as unusual patterns of transactions or access to sensitive data.
- **Website navigation:** GSP mining can be used to analyze website navigation patterns, such as the sequence of pages visited by users, and identify areas of the website that are frequently accessed or ignored.
- **Sensor data analysis**: GSP mining can be used to analyze sensor data, such as data from IoT devices, and identify patterns in the data that are indicative of certain conditions or states.

Example:



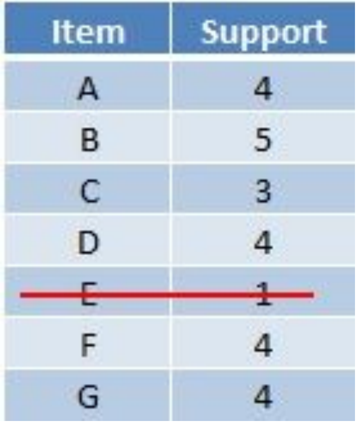
Transaction Date	Customer ID	Items Purchased
1	01	A
1	02	B
1	03	B
2	04	F
3	01	B
3	05	A
4	02	G
4	05	BC
5	03	F
6	04	AB
6	02	D
7	01	FG
7	05	G
8	04	C
8	03	G
9	05	F
9	01	C
9	03	AB
10	01	D
10	05	DE
10	04	D

- Suppose that the transaction data represents the day of the month that the item was purchased.
- As a human, we can see some repeat customers, and make some guesses about some of the repeat patterns that might exist, but we want an automated approach.
- An easy way to find sequential patterns is to utilize a modified process similar to the one developed for the Aprioi Algorithm that is for single purchases.
- To do this we just sort the table first by customer ID, and then by transaction time stamp.



Transaction Date	Customer ID	Items Purchased	Customer Sequence
1	01	A	<AB(FG)CD>
3	01	B	
7	01	FG	
9	01	C	
10	01	D	
1	02	B	<BGD>
4	02	G	
6	02	D	
1	03	B	<BFG(AB)>
5	03	F	
8	03	G	
9	03	AB	
2	04	F	<F(AB)CD>
6	04	AB	
8	04	C	
10	04	D	
3	05	A	<A(BC)GF(DE)>
4	05	BC	
7	05	G	
9	05	F	
10	05	DE	

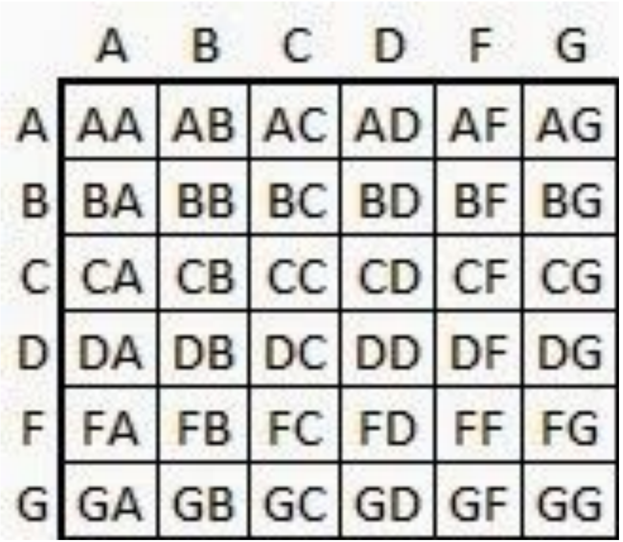
We'll use the data and say that the minimum support is 2 in this case, because items which had been brought on;y once are of no use and will have no sequence. If that's true we get the table below.



Item	Support
A	4
B	5
C	3
D	4
~~E~~	~~1~~
F	4
G	4

If you observe here, the support for B is listed at 5, not 6, even though the 3rd customer bought item B twice. This practice is also borrowed from the Apriori algorithm. In the Apriori algorithm, if a customer buys 2 candy bars at once, then we only count 1 candy bar when calculating the support, because we count transactions. The same applies here, except instead of counting how many transactions contain an item, or itemset, we are counting how many customers have an item/sequence.

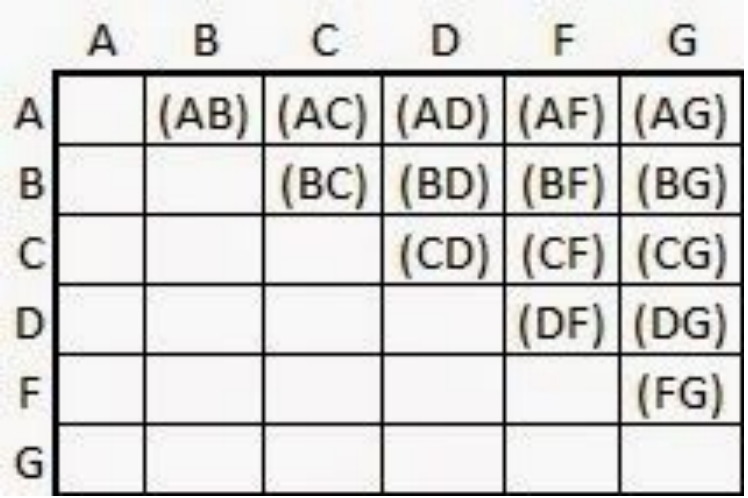
Once we have these items we start the iterative process of generating larger patterns from these items and checking if they have support.



	A	B	C	D	F	G
A	AA	AB	AC	AD	AF	AG
B	BA	BB	BC	BD	BF	BG
C	CA	CB	CC	CD	CF	CG
D	DA	DB	DC	DD	DF	DG
F	FA	FB	FC	FD	FF	FG
G	GA	GB	GC	GD	GF	GG

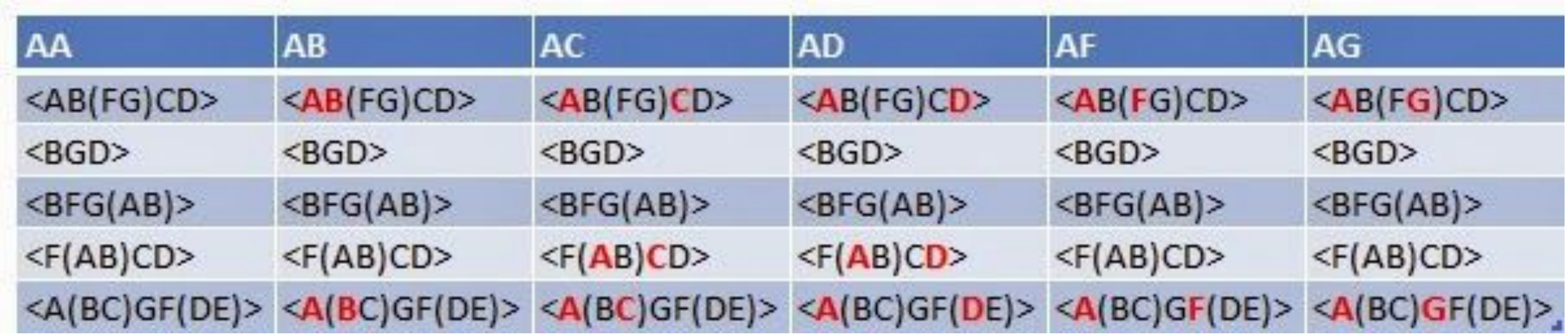
Because order matters, there are a lot more options. Oh and in case you forgot, there is also the possibility of 2+ items being bought in the same transaction.

The diagonal values are blank here because they're already covered in the first 2-item table. Just to make sure you understand, (AA) isn't there because if A and A were bought in the same transaction it would only be counted once so we never have two of the same item together within a parentheses. The lower left is also blank, but this is because of the convention I already described of always listing items purchased together in ascending order.



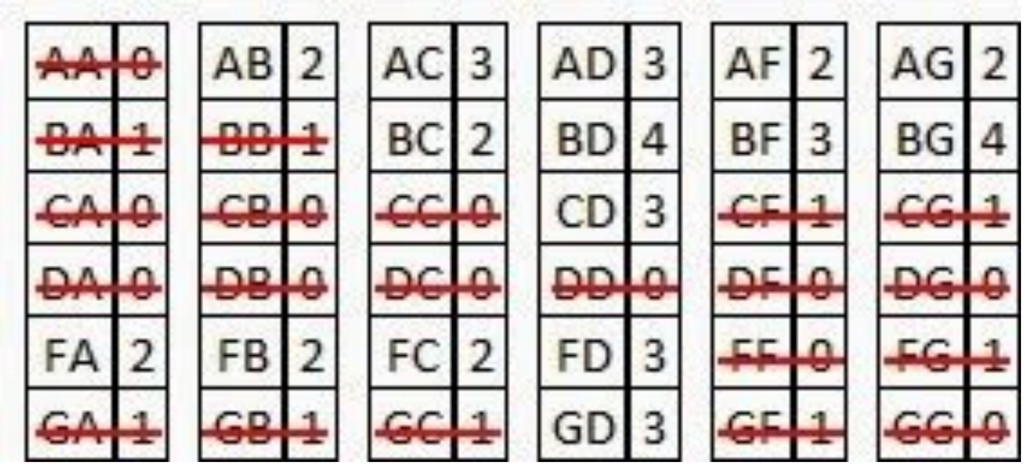
	A	B	C	D	F	G
A		(AB)	(AC)	(AD)	(AF)	(AG)
B			(BC)	(BD)	(BF)	(BG)
C				(CD)	(CF)	(CG)
D					(DF)	(DG)
F						(FG)
G						

So, now that we understand all of that, we can check our 51 possible 2-item sequences against the database to see if they meet the support threshold.



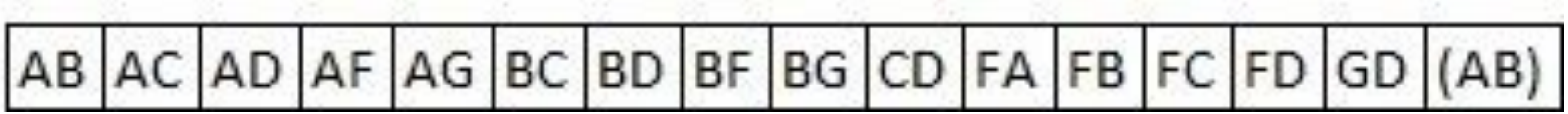
AA	AB	AC	AD	AF	AG
<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>
<BGD>	<BGD>	<BGD>	<BGD>	<BGD>	<BGD>
<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>
<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>
<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>

- In the first column, we see that there are no customers that ever bought item A on more than 1 occasion, so support there is 0. For <AB>, you can see that customers 1 and 5 bought those items in that sequence.
- For <AC>, notice that customer 4 bought (AB) together then C, but we get to pick out just A from that combined transaction to get our <AC> sequence.
- Notice that the items don't have to be right next to each other. It's OK to have other items in between. <AD> follows similar logic. For <AF> we don't count customers 3 or 4 because although they have A and F, they aren't in the right order. To be counted here, they need to be A, then F.
- When we look for the support of <FA> those will get counted. There is one last rule you need to know for counting support of sequences. If you are looking for support of a sequence like <FG>, then customer 1 doesn't count. It only counts if F is bought before G, NOT at the same time.
- If you work through the rest of the 2-item sequences that weren't purchased together (i.e. not ones like (AB)), you get support counts that look like this



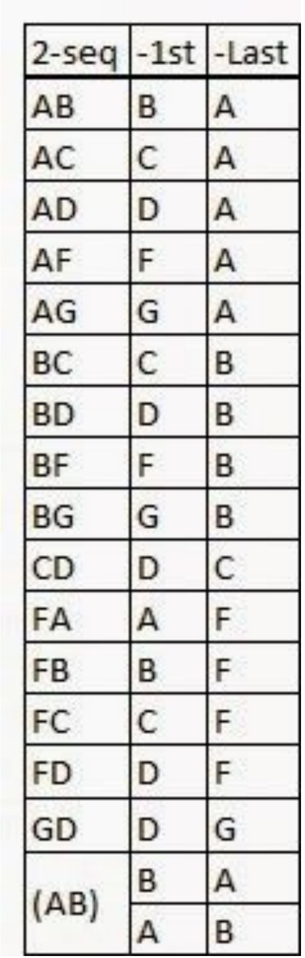
~~AA~~	~~0~~	AB	2	AC	3	AD	3	AF	2	AG	2
~~BA~~	~~1~~	~~BB~~	~~1~~	BC	2	BD	4	BF	3	BG	4
~~CA~~	~~0~~	~~CB~~	~~0~~	~~CC~~	~~0~~	CD	3	~~CF~~	~~1~~	~~CG~~	~~1~~
~~DA~~	~~0~~	~~DB~~	~~0~~	~~DC~~	~~0~~	~~DD~~	~~0~~	~~DF~~	~~0~~	~~DG~~	~~0~~
FA	2	FB	2	FC	2	FD	3	~~FF~~	~~0~~	~~FG~~	~~1~~
~~GA~~	~~1~~	~~GB~~	~~1~~	~~GC~~	~~1~~	GD	3	~~GF~~	~~1~~	~~GG~~	~~0~~

You see that I have struck out any 2-item sequence that do not meet the support threshold of 2. Now let's take a look at the 2-item sets that could have been purchased together (e.g. (AB)). This one is a little bit easier, because there are only 5 times in our database when 2 items were purchased at the same time. They were (FG), (AB), (AB), (BC) and (DE). Only (AB) shows up twice so we keep it and get rid of the rest. Now we have these 2-item sequences left to work with:



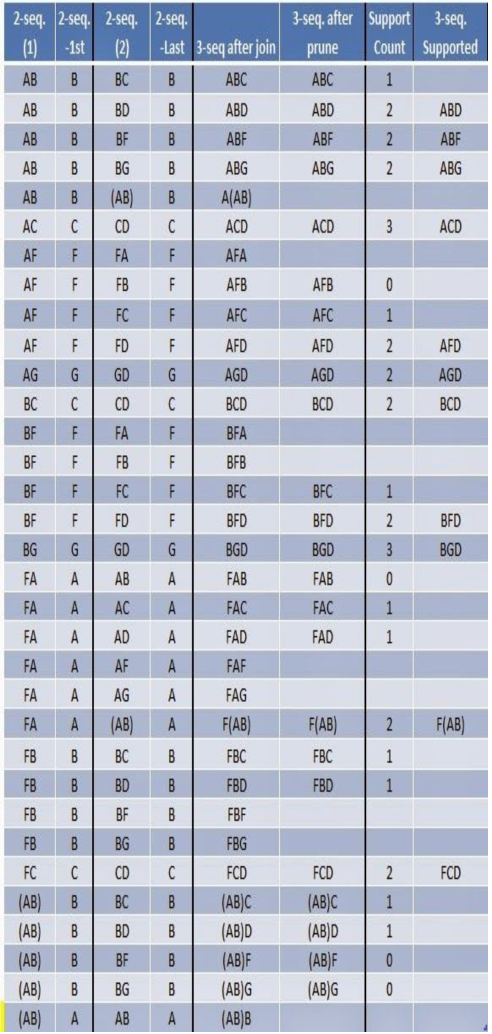
AB	AC	AD	AF	AG	BC	BD	BF	BG	CD	FA	FB	FC	FD	GD	(AB)

- Now we start the next iteration looking for 3-item sequences.
- You take all of the remaining sequences from the last step (AB, AC, AD, AF, AG, BC...), then you remove the first item from the sequence.
- For AB, we remove A and we're left with B. Do this for all of the sequences (see 2nd column in table).
- Then you do the same thing, but remove the last item (3rd column in table) .
- So if you're removing the "first" item from (AB) you remove A and get B leftover, and then you remove B and get A leftover.

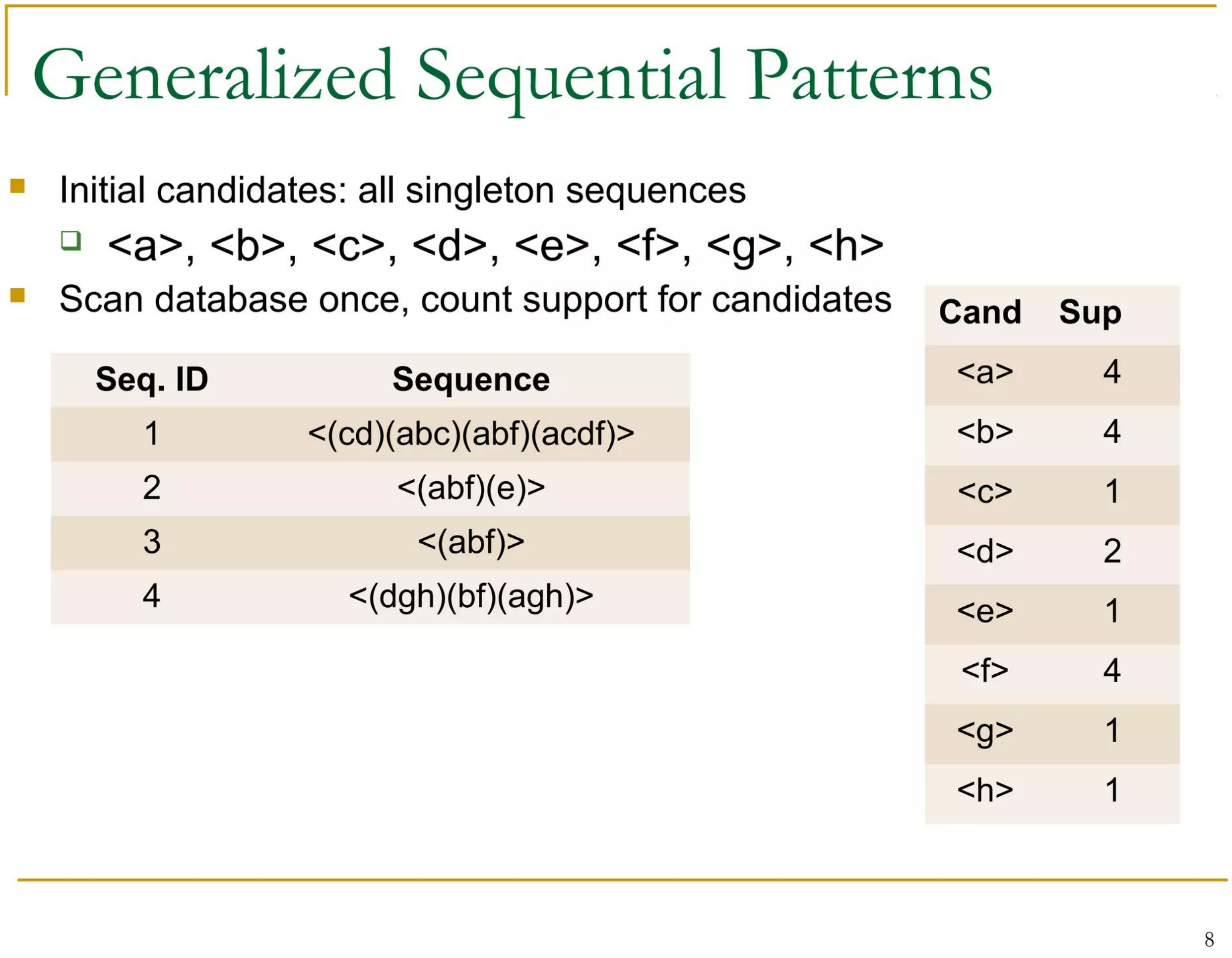


2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

- Now that we have this information ,We combine the sequences together where their "-1st" and "-Last" columns match.
- So if we're starting from the top, we see that the 2-item sequence AB matches up with BC, BD, BF, BG and (AB) to create ABC, ABD, ABF, ABG and A(AB).
- The A(AB) one is tricky because you have to remind yourself that the order within the parentheses is just convention and it could easily be written A(BA) which makes it easier to see that the B's match up.
- Working through the rest of the table this way (being careful not to create any duplicate candidates) we populate the "3-seq after join" column in the following table.



2-seq. (1)	2-seq. -1st	2-seq. (2)	2-seq. -Last	3-seq after join	3-seq. after prune	Support Count	3-seq. Supported
AB	B	BC	B	ABC	ABC	1	
AB	B	BD	B	ABD	ABD	2	ABD
AB	B	BF	B	ABF	ABF	2	ABF
AB	B	BG	B	ABG	ABG	2	ABG
AB	B	(AB)	B	A(AB)			
AC	C	CD	C	ACD	ACD	3	ACD
AF	F	FA	F	AFA			
AF	F	FB	F	AFB	AFB	0	
AF	F	FC	F	AFC	AFC	1	
AF	F	FD	F	AFD	AFD	2	AFD
AG	G	GD	G	AGD	AGD	2	AGD
BC	C	CD	C	BCD	BCD	2	BCD
BF	F	FA	F	BFA			
BF	F	FB	F	BFB			
BF	F	FC	F	BFC	BFC	1	
BF	F	FD	F	BFD	BFD	2	BFD
BG	G	GD	G	BGD	BGD	3	BGD
FA	A	AB	A	FAB	FAB	0	
FA	A	AC	A	FAC	FAC	1	
FA	A	AD	A	FAD	FAD	1	
FA	A	AF	A	FAF			
FA	A	AG	A	FAG			
FA	A	(AB)	A	F(AB)	F(AB)	2	F(AB)
FB	B	BC	B	FBC	FBC	1	
FB	B	BD	B	FBD	FBD	1	
FB	B	BF	B	FBF			
FB	B	BG	B	FBG			
FC	C	CD	C	FCD	FCD	2	FCD
(AB)	B	BC	B	(AB)C	(AB)C	1	
(AB)	B	BD	B	(AB)D	(AB)D	1	
(AB)	B	BF	B	(AB)F	(AB)F	0	
(AB)	B	BG	B	(AB)G	(AB)G	0	
(AB)	A	AB	A	(AB)B			

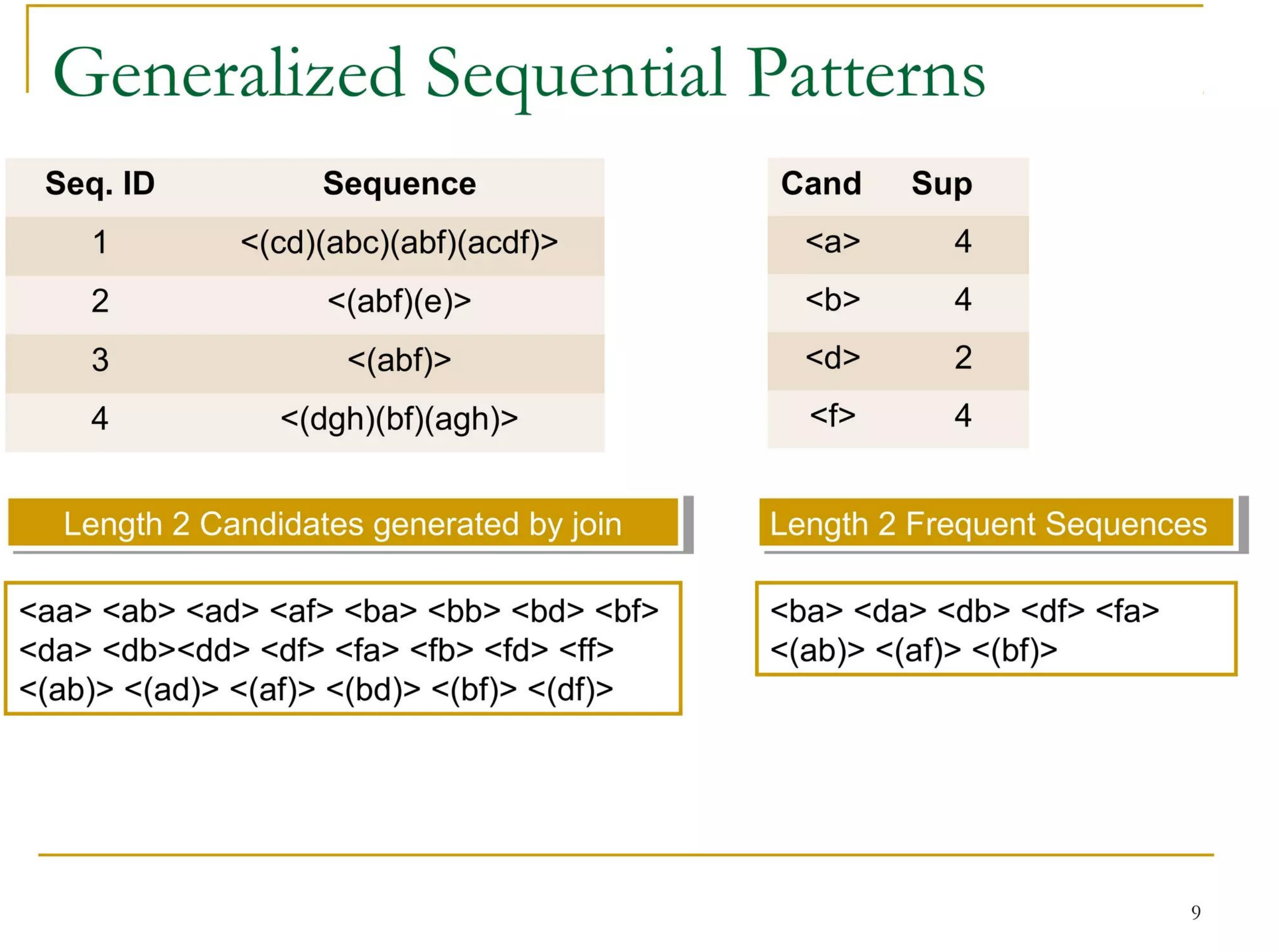


Generalized Sequential Patterns

- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

Seq. ID	Sequence
1	<(cd)(abc)(abf)(acdf)>
2	<(abf)(e)>
3	<(abf)>
4	<(dgh)(bf)(agh)>

Cand	Sup
<a>	4
<b>	4
<c>	1
<d>	2
<e>	1
<f>	4
<g>	1
<h>	1



Generalized Sequential Patterns

Seq. ID	Sequence
1	<(cd)(abc)(abf)(acdf)>
2	<(abf)(e)>
3	<(abf)>
4	<(dgh)(bf)(agh)>

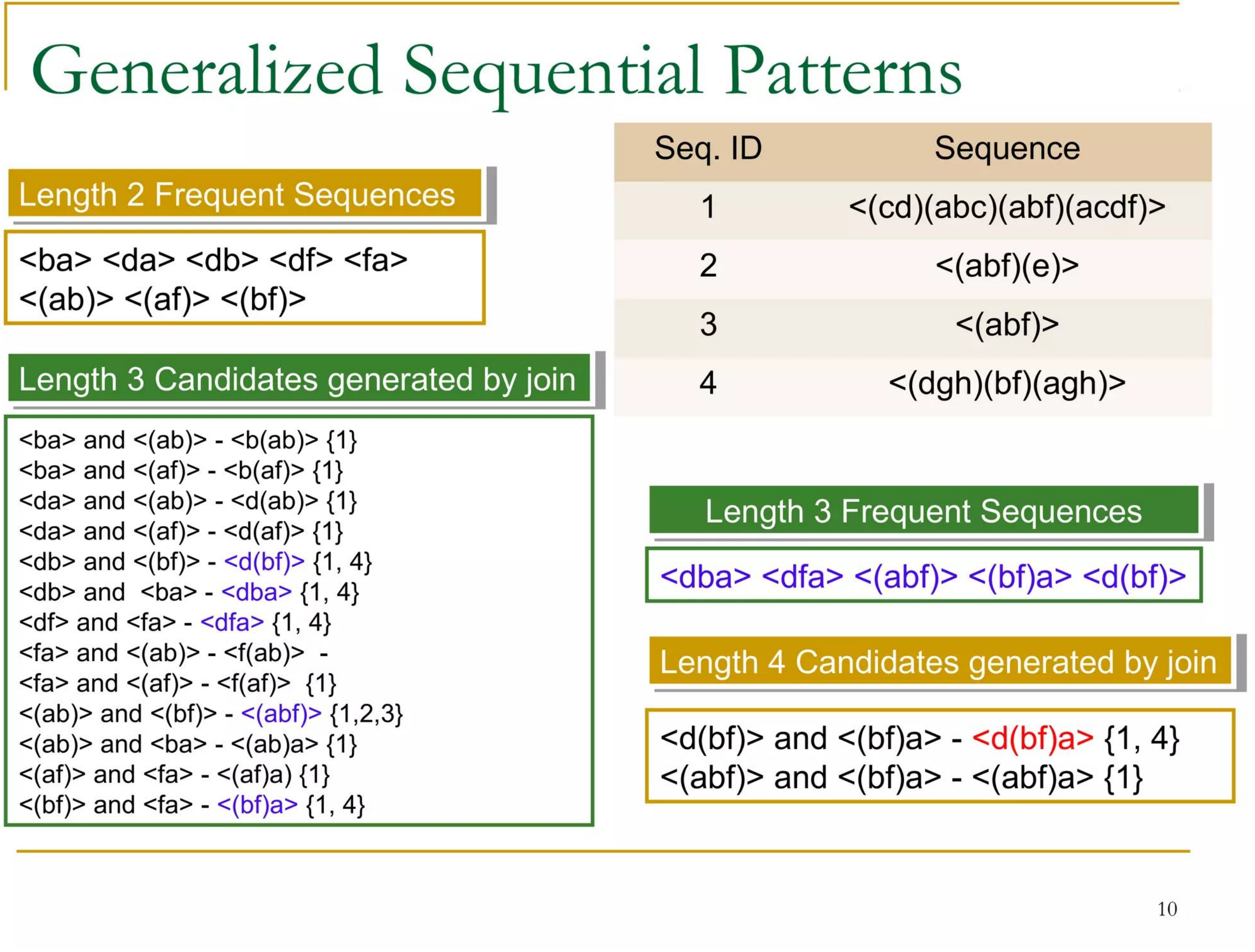
Cand	Sup
<a>	4
<b>	4
<d>	2
<f>	4

Length 2 Candidates generated by join

<aa> <ab> <ad> <af> <ba> <bb> <bd> <bf>
<da> <db><dd> <df> <fa> <fb> <fd> <ff>
<(ab)> <(ad)> <(af)> <(bd)> <(bf)> <(df)>

Length 2 Frequent Sequences

<ba> <da> <db> <df> <fa>
<(ab)> <(af)> <(bf)>



Generalized Sequential Patterns

Seq. ID	Sequence
1	<(cd)(abc)(abf)(acdf)>
2	<(abf)(e)>
3	<(abf)>
4	<(dgh)(bf)(agh)>

Length 2 Frequent Sequences

<ba> <da> <db> <df> <fa>
<(ab)> <(af)> <(bf)>

Length 3 Candidates generated by join

<ba> and <(ab)> - <b(ab)> {1}
<ba> and <(af)> - <b(af)> {1}
<da> and <(ab)> - <d(ab)> {1}
<da> and <(af)> - <d(af)> {1}
<db> and <(bf)> - <d(bf)> {1, 4}
<db> and <ba> - <dba> {1, 4}
<df> and <fa> - <dfa> {1, 4}
<fa> and <(ab)> - <f(ab)> -
<fa> and <(af)> - <f(af)> {1}
<(ab)> and <(bf)> - <(abf)> {1,2,3}
<(ab)> and <ba> - <(ab)a> {1}
<(af)> and <fa> - <(af)a) {1}
<(bf)> and <fa> - <(bf)a> {1, 4}

Length 3 Frequent Sequences

<dba> <dfa> <(abf)> <(bf)a> <d(bf)>

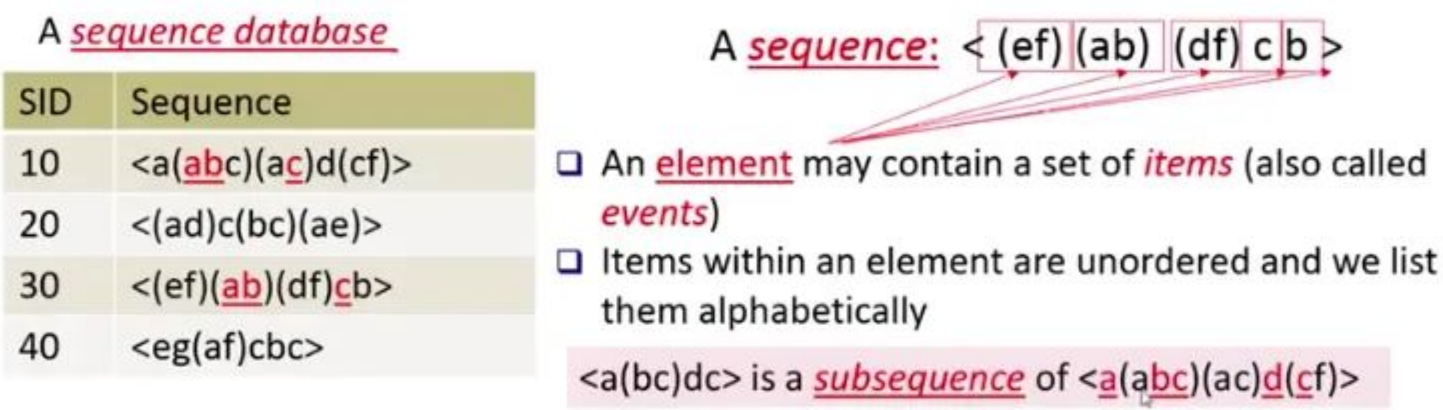
Length 4 Candidates generated by join

<d(bf)> and <(bf)a> - <d(bf)a> {1, 4}
<(abf)> and <(bf)a> - <(abf)a> {1}

Sequence Pattern Discovery using Equivalence Class (SPADE)

What is SPADE? Vertical Based Data

- An algorithm to Frequent Sequence Mining is the SPADE (Sequential PAttern Discovery using Equivalence classes) algorithm. It uses a **vertical id-list database** format, where we associate to each sequence a list of objects in which it occurs.
- This Sequence Pattern Mining algorithm identifies each element in each sequence in a dataset with a Sequence ID (SID) and the Element ID (EID). Candidates of length 1 are constructed, and the SIDs and EIDs of all elements where they occur are noted.
- Candidates of higher length are constructed with the property that the Element IDs of all the elements in the candidate should be in increasing order. This algorithm also facilitates joins (for example, if a set of SIDs and EIDs are identified for candidates ab and ba, then SIDs and EIDs can be obtained for candidate aba through joins).



A *sequence database*

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

A ***sequence***: < (ef) (ab) (df) c b >

- ❑ An element may contain a set of *items* (also called *events*)
- ❑ Items within an element are unordered and we list them alphabetically

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Sequential Pattern Mining in Vertical Data Format: The SPADE Algorithm

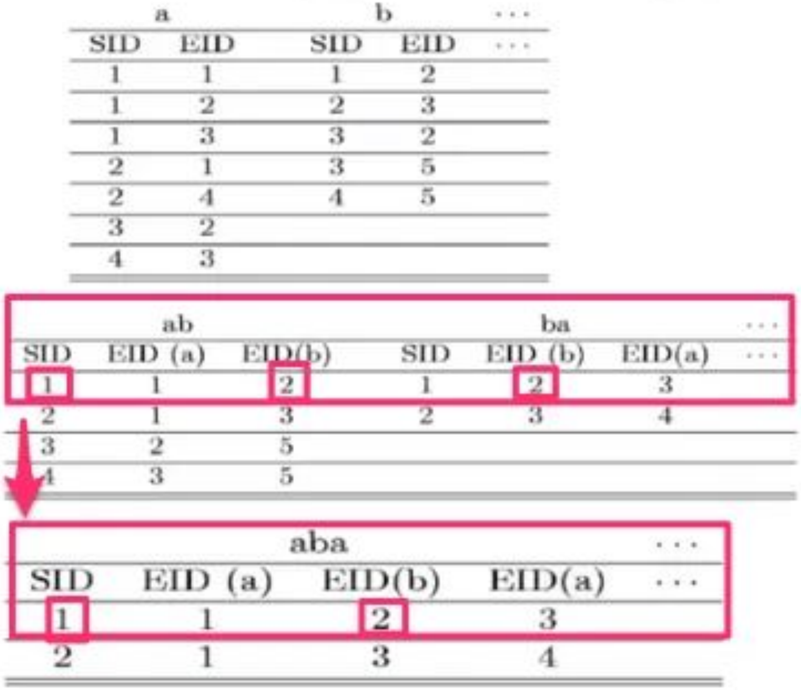
- A sequence database is mapped to: <SID, EID>
- Grow the subsequences (patterns) one item at a time by Apriori candidate generation

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

min_sup = 2

Ref: SPADE (Sequential PAttern Discovery using Equivalent Class) [M. Zaki 2001]

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c



a		b		···
SID	EID	SID	EID	···
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

ab			ba			···
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	···
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				···
SID	EID (a)	EID(b)	EID(a)	···
1	1	2	3	
2	1	3	4	

Calculate the <SID,EID> of each element of length one.

Combine frequent sequence of length two:

    If the SID of both the elements are same, and
    The EID of first element is less than the second element.

Join the frequent sequence of length three:

    The SID of sequence (of length 2) is same
    The EID of sequence (of length 1) is same.

Time Series Analysis

What is Time Series Analysis?

Time series analysis is a specific way of analyzing a sequence of data points collected over time. In TSA, analysts record data points at consistent intervals over a set period rather than just recording the data points intermittently or randomly.

Components of Time Series Analysis

- **Trend:** In which there is no fixed interval and any divergence within the given dataset is a continuous timeline. The trend would be Negative or Positive or Null Trend
- **Seasonality**: In which regular or fixed interval shifts within the dataset in a continuous timeline. Would be bell curve or saw tooth
- **Cyclical**: In which there is no fixed interval, uncertainty in movement and its pattern
- **Irregularity:** Unexpected situations/events/scenarios and spikes in a short time span.

Trend

The trend shows the general tendency of the data to increase or decrease during a long period of time. A trend is a smooth, general, long-term, average tendency. It is not always necessary that the increase or decrease is in the same direction throughout the given period of time.

It is observable that the tendencies may increase, decrease or are stable in different sections of time. But the overall trend must be upward, downward or stable. The population, agricultural production, items manufactured, number of births and deaths, number of industry or any factory, number of schools or colleges are some of its example showing some kind of tendencies of movement.

Linear and Non-Linear Trend

If we plot the time series values on a graph in accordance with time t. The pattern of the data clustering shows the type of trend. If the set of data cluster more or less round a straight line, then the trend is linear otherwise it is non-linear (Curvilinear).

Seasonality

These are the rhythmic forces which operate in a regular and periodic manner over a span of less than a year. They have the same or almost the same pattern during a period of 12 months. This variation will be present in a time series if the data are recorded hourly, daily, weekly, quarterly, or monthly.

These variations come into play either because of the natural forces or man-made conventions. The various seasons or climatic conditions play an important role in seasonal variations. Such as production of crops depends on seasons, the sale of umbrella and raincoats in the rainy season, and the sale of electric fans and A.C. shoots up in summer seasons.

Cyclical

The variations in a time series which operate themselves over a span of more than one year are the cyclic variations. This oscillatory movement has a period of oscillation of more than a year. One complete period is a cycle. This cyclic movement is sometimes called the ‘Business Cycle’.

It is a four-phase cycle comprising of the phases of prosperity, recession, depression, and recovery. The cyclic variation may be regular are not periodic. The upswings and the downswings in business depend upon the joint nature of the economic forces and the interaction between them.

Irregular Variations

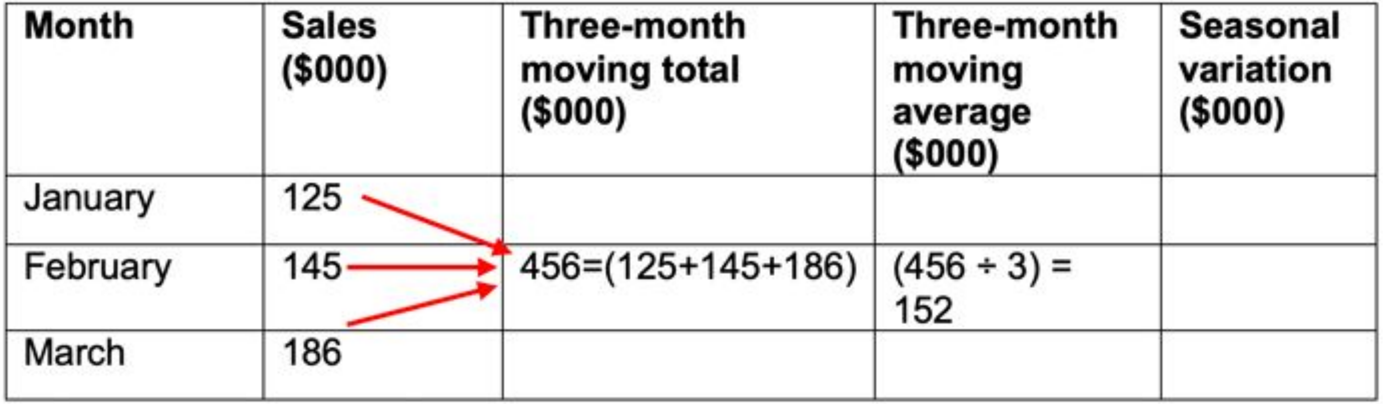
It refers to variations that are uncontrollable and inevitable. It occurs randomly, opposite to regular changes or occurrences, and does not associate with a pattern. These fluctuations are unpredictable and unexplainable. Forces like natural and man-made disasters can trigger irregular variations.

Forecasting by Time series analysis:

- Time series forecasting means assessing the time-stamped data using statistical calculations and modeling to make predictions and induce strong strategic decision-making. The process is widely adopted in many sectors, for example, sales forecasting and weather forecasting. Forecasting highly depends on the nature of the data, and the process is usually performed on historical data.
- Examples of time series methods used for forecasting are
  - **Autoregression (AR),**
  - **Moving Average (MA),**
  - **Autoregressive Moving Average (ARMA), and**
  - **Autoregressive Integrated Moving Average (ARIMA)**: The ARIMA model aims to explain data by using time series data on its past values and uses linear regression to make predictions.

Example: Calculate the three-month moving average.

- Add together the first three sets of data, for this example it would be January, February and March.
- This gives a total of (125+145+186) = 456. Put this total in the middle of the data you are adding, so in this case across from February. Then calculate the average of this total, by dividing this figure by 3 (the figure you divide by will be the same as the number of time periods you have added in your total column).
- Our three-month moving average is therefore (456 ÷ 3) = 152.



Month	Sales ($000)	Three-month moving total ($000)	Three-month moving average ($000)	Seasonal variation ($000)
January	125			
February	145	456=(125+145+186)	(456 ÷ 3) = 152	
March	186			

Month	Sales ($000)	Three-month moving total ($000)	Three-month moving average ($000)	Seasonal variation ($000)
January	125			
February	145	456=(125+145+186)	(456 ÷ 3) = 152	
March	186	462=(145+186+131)	(462 ÷ 3) = 154	
April	131	468=(186+131+151)	(468 ÷ 3) = 156	
May	151	474	158	
June	192	480	160	
July	137	486	162	
August	157	492	164	
September	198	498	166	
October	143	504	168	
November	163	510	170	
December	204			

Step - 2: Calculate the trend

The three-month moving average represents the trend. From our example we can see a clear trend in that each moving average is $2,000 higher than the preceding month moving average. This suggests that the sales revenue for the company is, on average, growing at a rate of $2,000 per month.

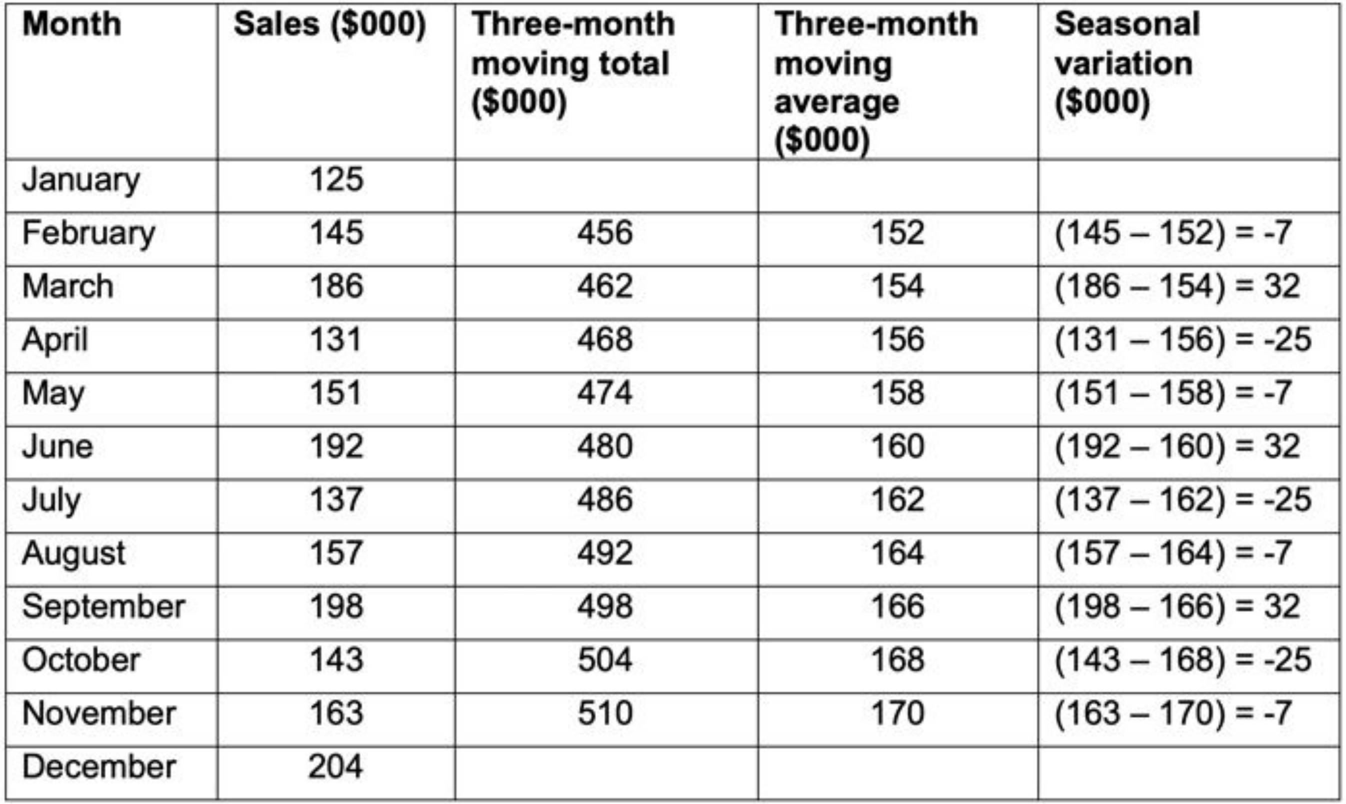
This trend can now be used to predict future underlying sales values.

Step 3 – Calculate the seasonal variation

Once a trend has been established, any seasonal variation can be calculated. The seasonal variation can be assumed to be the difference between the actual sales and the trend (three-month moving average) value. Seasonal variations can be calculated using the additive or multiplicative models.

Using the additive model:

To calculate the seasonal variation, go back to the table and for each average calculated, compare the average to the actual sales figure for that period.

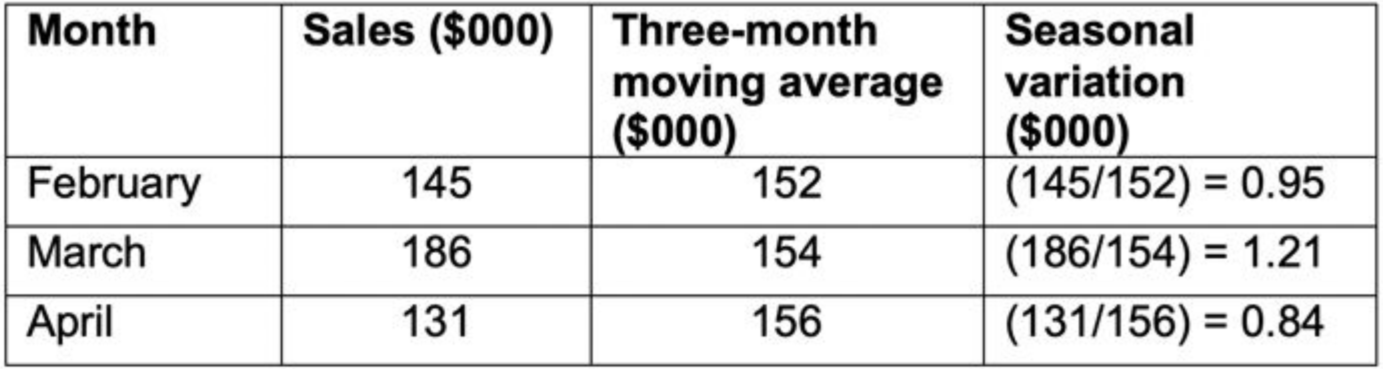


Month	Sales ($000)	Three-month moving total ($000)	Three-month moving average ($000)	Seasonal variation ($000)
January	125			
February	145	456	152	(145 – 152) = -7
March	186	462	154	(186 – 154) = 32
April	131	468	156	(131 – 156) = -25
May	151	474	158	(151 – 158) = -7
June	192	480	160	(192 – 160) = 32
July	137	486	162	(137 – 162) = -25
August	157	492	164	(157 – 164) = -7
September	198	498	166	(198 – 166) = 32
October	143	504	168	(143 – 168) = -25
November	163	510	170	(163 – 170) = -7
December	204			

- A negative variation means that the actual figure in that period is less than the trend and a positive figure means that the actual is more than the trend.
- From the data we can see a clear three-month cycle in the seasonal variation. Every first month has a variation of -7, suggesting that this month is usually $7,000 below the average.
- Every second month has a variation of 32 suggesting that this month is usually $32,000 above the average.
- In month 3, the variation suggests that every third month, the actual will be $25,000 below the average.
- It is assumed that this pattern of seasonal adjustment will be repeated for each three-month period going forward.

Using the multiplicative model:

If we had used the multiplicative model, the variations would have been expressed as a percentage of the average figure, rather than an absolute. For example:



Month	Sales ($000)	Three-month moving average ($000)	Seasonal variation ($000)
February	145	152	(145/152) = 0.95
March	186	154	(186/154) = 1.21
April	131	156	(131/156) = 0.84