TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Data Mining Seminar:
Chapter 8 (Mining Stream, Time-Series, and Sequence Data)

Content

I. Information Group

MSSV	Tên	Công việc	Đánh giá
19127587	Trương Chí Toàn	Mining Stream	Hoàn thành tốt công việc
19127630	Trần Quốc Việt	Time-Series	Hoàn thành tốt công việc
19127592	Lê Minh Trí	Sequence Data	Hoàn thành tốt công việc

II. Theory

A. Mining Stream

Tremendous and potentially infinite volumes of data streams are often generated by real-time surveillance systems, communication networks, Internet traffic, on-line transactions in the financial market or retail industry, electric power grids, industry production processes, scientific and engineering experiments, remote sensors, and other dynamic environments. Unlike traditional data sets, **stream data** flow in and out of a computer system continuously and with varying update rates. They are *temporally ordered*, f*ast changing*, *massive*, and *potentially infinite*. It may be impossible to store an entire data stream or to scan through it multiple times due to its tremendous volume. Moreover, stream data tend to be of a rather low level of abstraction, whereas most analysts are interested in relatively high-level dynamic changes, such as trends and deviations.

1.1. Methodologies for Stream Data Processing and Stream Data Systems

For effective processing of stream data, new data structures, techniques, and algorithms are needed. Because we do not have an infinite amount of space to store stream data, we often trade off between accuracy and storage. That is, we generally are willing to settle for approximate rather than exact answers. **Synopses** allow for this by providing summaries of the data, which typically can be used to return approximate answers to queries. Synopses use synopsis data structures, which are any data structures that are substantially smaller than their base data set (in this case, the stream data).

In this section, we examine some common synopsis data structures and techniques.

Random Sampling

Rather than deal with an entire data stream, we can think of sampling the stream at periodic intervals.

Sliding Windows

Instead of sampling the data stream randomly, we can use the sliding window model to analyze stream data. The basic idea is that rather than running computations on all of the data seen so far, or on some sample, we can make decisions based only on recent data

Histograms

The histogram is a synopsis data structure that can be used to approximate the frequency distribution of element values in a data stream. A histogram partitions the data into a set of contiguous buckets. Depending on the partitioning rule used, the width (bucket value range) and depth (number of elements per bucket) can vary.

Multiresolution Methods

A common way to deal with a large amount of data is through the use of data reduction methods. A popular data reduction method is the use of divide-and conquer strategies such as multiresolution data structures. These allow a program to trade off between accuracy and storage, but also offer the ability to understand a data stream at multiple levels of detail.

Sketches

From a database perspective, sketch partitioning was developed to improve the performance of sketching on data stream query optimization. Sketch partitioning uses coarse statistical information on the base data to intelligently partition the domain of the underlying attributes in a way that provably tightens the error guarantees.

Randomized Algorithms

Randomized algorithms, in the form of random sampling and sketching, are often used to deal with massive, high-dimensional data streams. The use of randomization often leads to simpler and more efficient algorithms in comparison to known deterministic algorithms.

1.2. Stream OLAP and Stream Data Cubes

A power supply station generates infinite streams of power usage data. Suppose individual user, street address, and second are the attributes at the lowest level of granularity. Given a large number of users, it is only realistic to analyze the fluctuation of power usage at certain high levels, such as by city or street district and by quarter (of an hour), making timely power supply adjustments and handling unusual situations.

Conceptually, for multidimensional analysis, we can view such stream data as a virtual data cube, consisting of one or a few measures and a set of dimensions, including one time dimension, and a few other dimensions, such as location, user-category, and so on. However, in practice, it is impossible to materialize such a data cube, because the materialization requires a huge amount of data to be computed and stored. Some efficient methods must be developed for systematic analysis of such data.

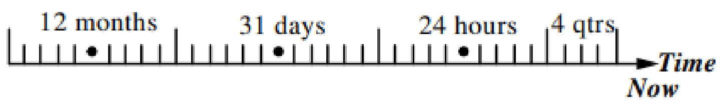
A realistic design is to explore several data compression techniques, including:

(1) **tilted time frame** on the time dimension

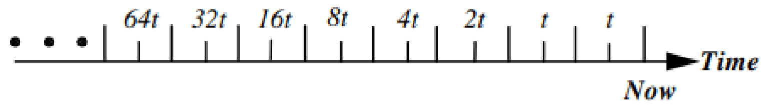
(2) storing data only at some **critical layers**

(3) exploring efficient computation of a very **partially materialized data cube.**

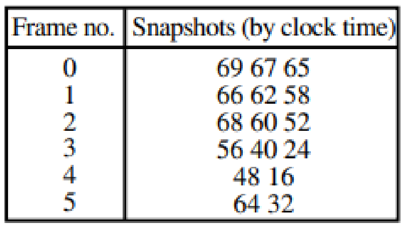
1.2.1 Time Dimension with Compressed Time Scale: Tilted Time Frame



(a) A natural tilted time frame model.



(b) A logarithmic tilted time frame model.



Frame no.	Snapshots (by clock time)
0	69 67 65
1	66 62 58
2	68 60 52
3	56 40 24
4	48 16
5	64 32

(c) A progressive logarithmic tilted time frame table.

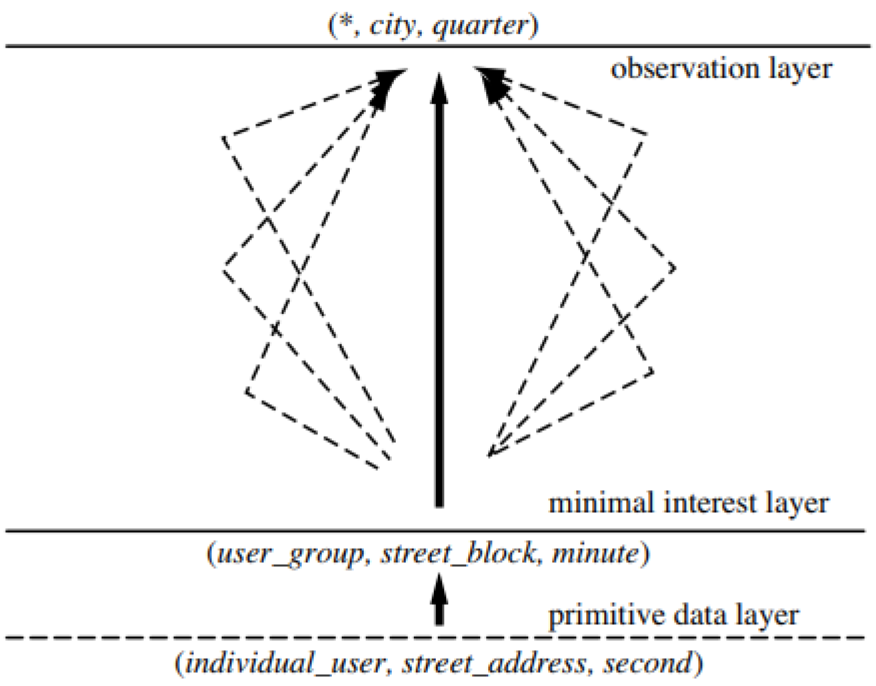
- A **natural tilted time frame model**, where the time frame (or window) is structured in multiple granularities based on the “natural” or usual time scale: the most recent 4 quarters (15 minutes), followed by the last 24 hours, then 31 days, and then 12 months.
- A **logarithmic tilted time frame model**, where the time frame is structured in multiple granularities according to a logarithmic scale.
- A **progressive logarithmic tilted time frame model**, where snapshots are stored at differing levels of granularity depending on the recency.

1.2.2 Critical Layers

In many applications, it is beneficial to dynamically and incrementally compute and store two critical cuboids (or layers), which are determined based on their conceptual and computational importance in stream data analysis.

The first layer, called the minimal interest layer, is the minimally interesting layer that an analyst would like to study.

The second layer, called the observation layer, is the layer at which an analyst (or an automated system) would like to continuously study the data.



1.3. Frequent-Pattern Mining in Data Streams

A pattern is considered frequent if its count satisfies a minimum support.
However, mining such patterns in dynamic data streams poses substantial new challenges.

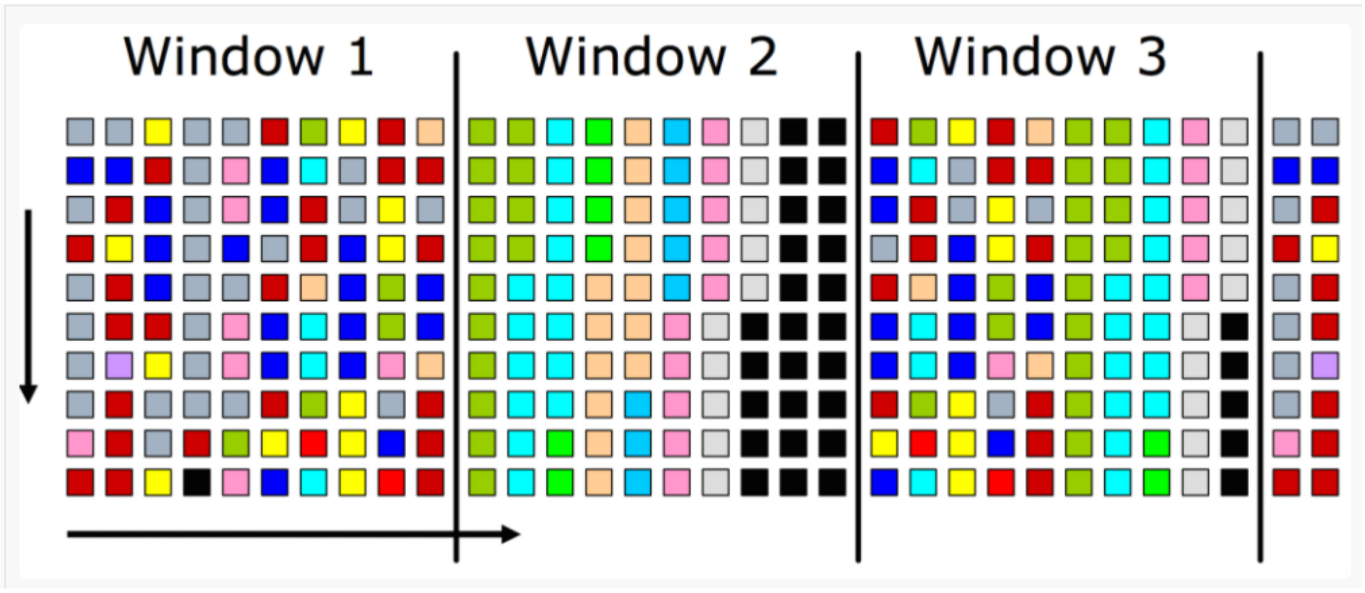
To overcome this difficulty, there are two possible approaches:

- One is to keep track of only a predefined, limited set of items and itemsets.
- The second approach is to derive an approximate set of answers.

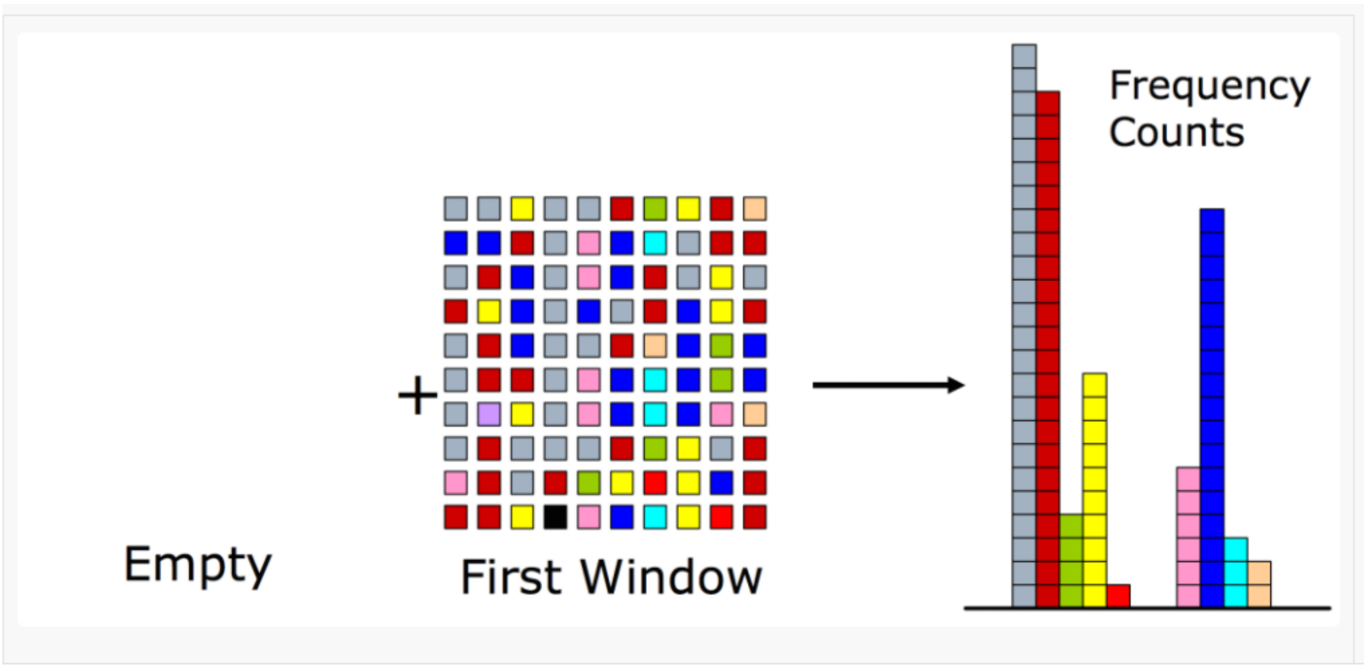
1.3.1 Lossy Counting Algorithm

The **lossy count algorithm** is an algorithm to identify elements in a data stream whose frequency count exceed a user-given threshold.

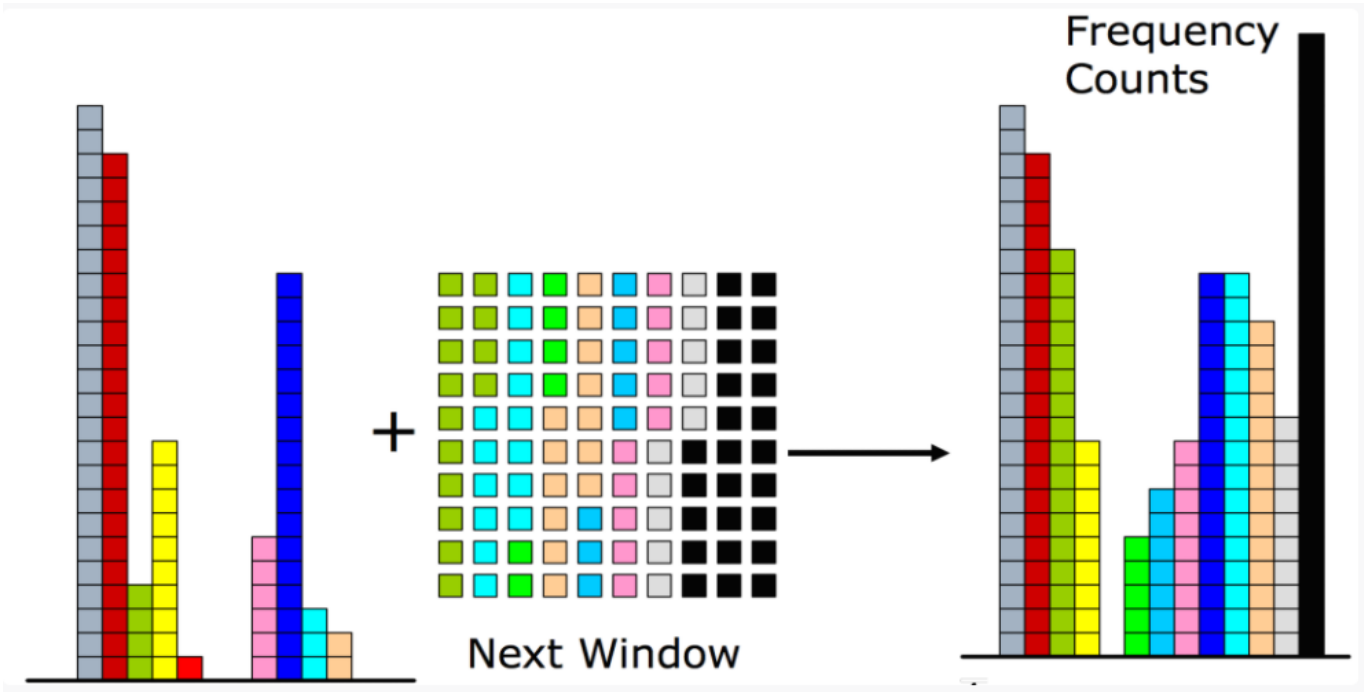
Step 1: Divide the incoming stream data into windows



Step 2: Increment the frequency count of each item according to the new bucket values. After each window, decrement all counters by 1.



Step 3: Repeat - Update counters and after each windows, decrement all counters by 1



1.4. Classification of Dynamic Data Streams

Several methods have been proposed for classification of stream data. We introduce three of them:

1) Hoefffing tree algorithm
2) Very Fast Decision Tree(VFDT)
3) Concept-adapting Very Fast Decision Tree

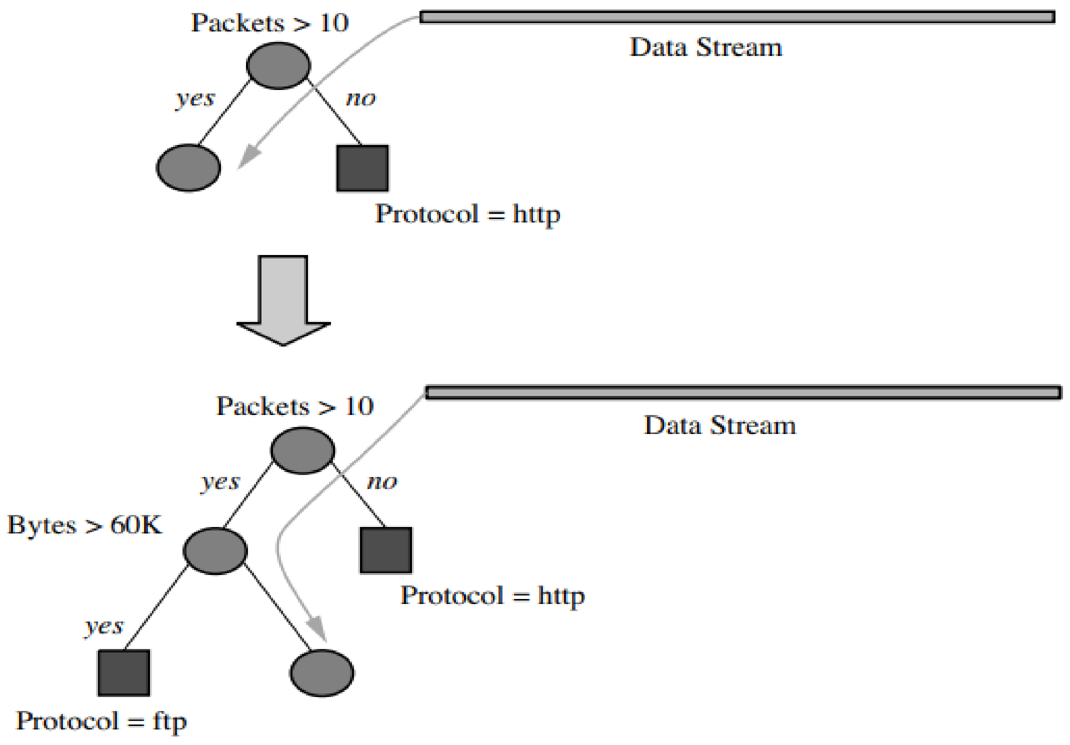
Hoeffding Tree Algorithm

The **Hoeffding tree algorithm** is a decision tree learning method for stream data classification. It was initially used to track Web clickstreams and construct models to predict which Web hosts and Web sites a user is likely to access.

Algorithm 1 Hoeffding Tree Induction Algorithm.

```
1: Let $HT$ be a tree with a single leaf (the root)
2: for all training examples do
3:   Sort example into leaf $l$ using $HT$
4:   Update sufficient statistics in $l$
5:   Increment $n_l$, the number of examples seen at $l$
6:   if $n_l \; mod \; n_{min} = 0$ and examples seen at $l$ not all of same class then
7:     Compute $\overline{G}_l(X_i)$ for each attribute
8:     Let $X_a$ be attribute with highest $\overline{G}_l$
9:     Let $X_b$ be attribute with second-highest $\overline{G}_l$
10:    Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
11:    if $X_a \neq X_\emptyset$ and $(\overline{G}_l(X_a) - \overline{G}_l(X_b) > \epsilon$ or $\epsilon < \tau)$ then
12:      Replace $l$ with an internal node that splits on $X_a$
13:      for all branches of the split do
14:        Add a new leaf with initialized sufficient statistics
15:      end for
16:    end if
17:  end if
18: end for
```

The figure demonstrates how new examples are integrated into the tree as they stream in



Very Fast Decision Tree (VFDT) and Concept-adapting Very Fast Decision Tree (CVFDT)

VFDT makes several modifications to the Hoeffding tree algorithm to improve both speed and memory utilization. However, it still cannot handle concept drift in data streams.

To adapt to concept-drifting data streams, the VFDT algorithm was further developed into the **Concept-adapting Very Fast Decision Tree algorithm** (CVFDT). CVFDT also uses a sliding window approach; however, it does not construct a new model from scratch each time. Rather, it updates statistics at the nodes by incrementing the counts associated with new examples and decrementing the counts associated with old ones. Therefore, if there is a concept drift, some nodes may no longer pass the Hoeffding bound.

Empirical studies show that CVFDT achieves better accuracy than VFDT with time-changing data streams. In addition, the size of the tree in CVFDT is much smaller than that in VFDT, because the latter accumulates many outdated examples.

B. Time-Series

What is a time-series ?

Time series data is a collection of observations obtained through repeated measurements over time. The values are typically measured at equal time intervals (e.g., hourly, daily, weekly, yearly)

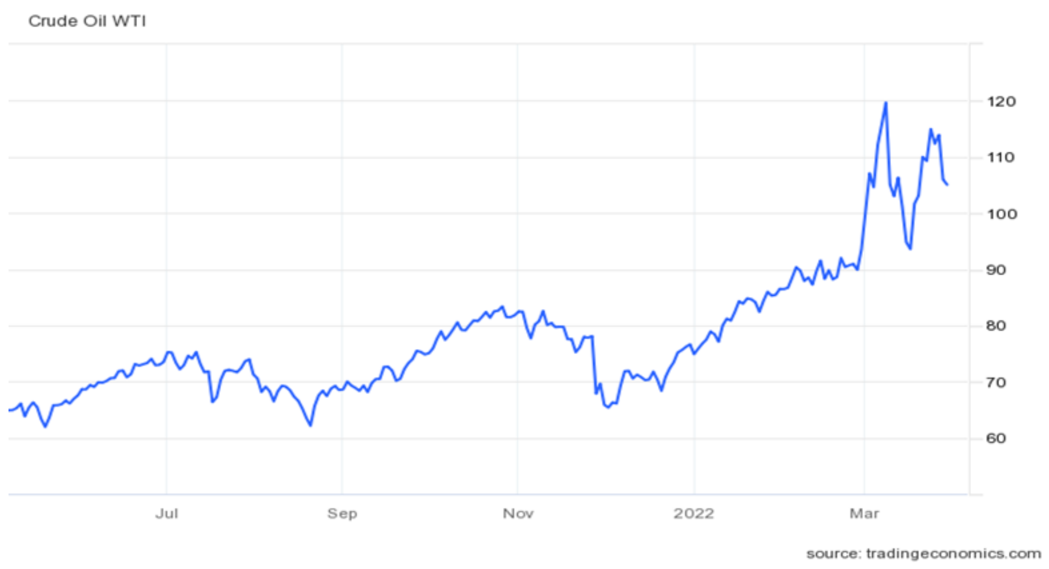
Nowadays, Time series data is everywhere. The evolution of sensors, mobile devices, data hundreds of gigabytes, terabytes per minute (data from satellites around the world), stock market,...

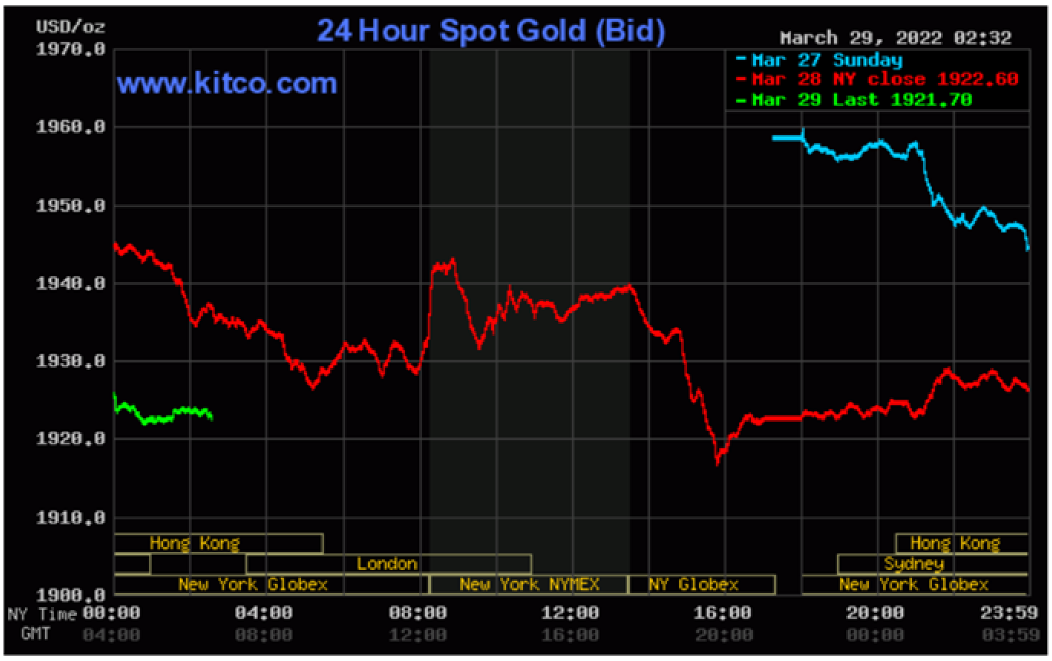
Time series data is often present in the field noise in real life
Examples of time series analysis:

- Observation of natural phenomena (such as atmosphere, temperature, wind, earthquake). Such as Daily Weather data, Sensors are located all over the earth to measure things like temperature, precipitation levels, wind direction, wind speed, and atmospheric pressure

- Stock market analysis, economic and sales forecasting, budgetary analysis, utility studies, inventory studies, yield projections

- Scientific and engineering experiments, and medical treatments





The issues that have been solved are:

1. How can we find correlation relationships within time-series data?

2. How can we analyze such huge numbers of time series to find similar or regular patterns, trends, bursts (such as sudden sharp changes), and outliers, with fast or even on-line real-time response?

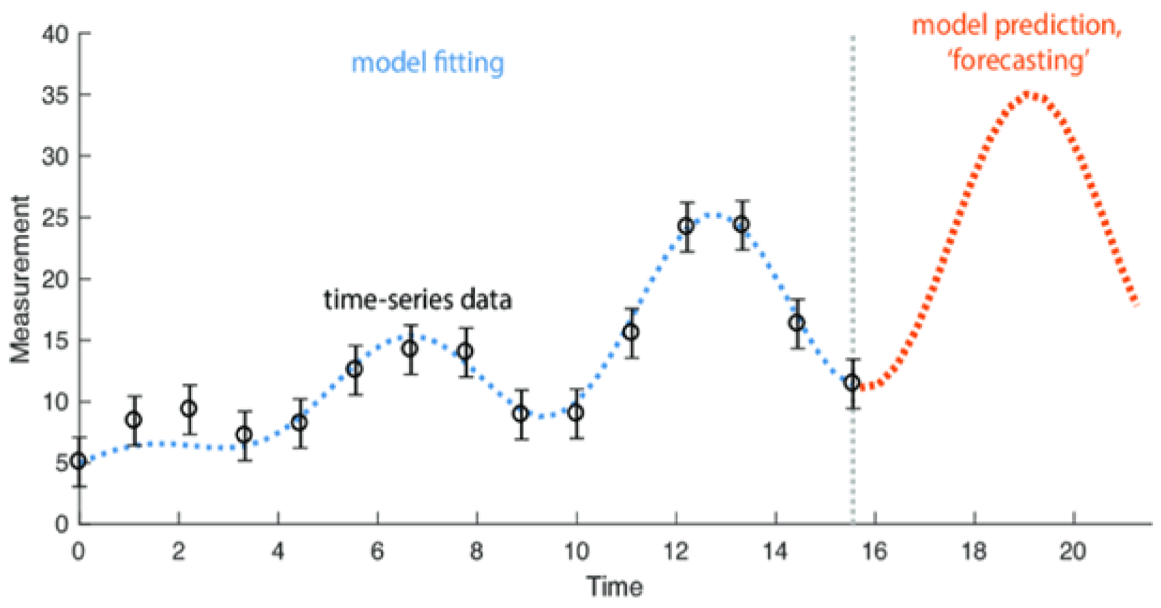
These two issues are necessary, but also it faces a difficult challenge. In the next section we will introduce two aspects of analysis. That is *trend analysis* and *similarity search*.

1. Trend analysis

- There are two goals in time-series analysis:

+ **Modeling time series** : this goal is to gain insight into the characteristics of data at the present time using mathematical statistical methods.

+ **Forecasting time series**: is the use a model to predict temporal events based on known past events, so predicting the future values of the time-series variables



❖ Components or movements for characterizing time-series data:

- Trend analysis consists of the following four major components or movements

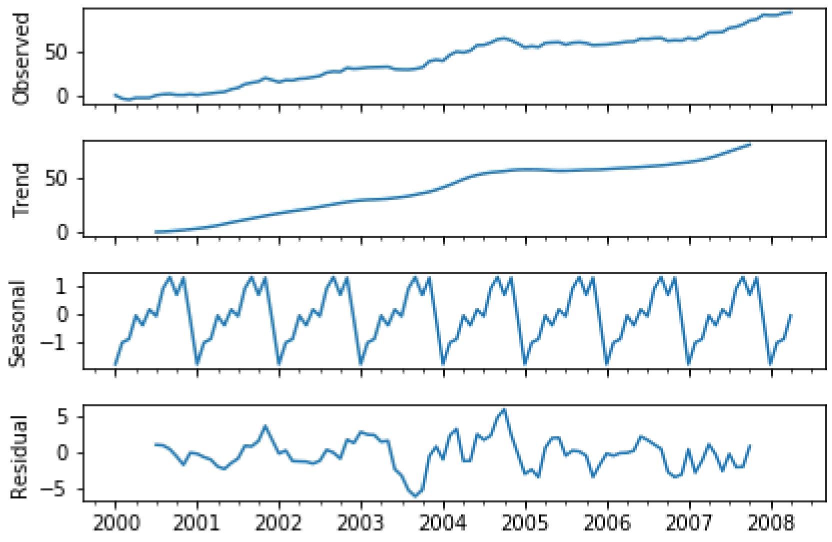
1. Trend or long-term movements: The trend shows the general tendency of the data to increase or decrease during a long period of time. A trend is a smooth, general, long-term, average tendency
2. Cyclic movements or cyclic variations: The variations in a time series which operate themselves over a span of more than one year are the cyclic variations. This oscillatory movement has a period of oscillation of more than a year.
3. Seasonal movements or seasonal variations: These are systematic or calendar related. Examples include events that recur annually, such as the sudden increase in sales of chocolates and flowers before Valentine's Day or of department store items before Christmas. The observed increase in water consumption in summer due to warm weather is another example. In these examples, seasonal movements are the identical or nearly identical patterns that a time series appears to follow during corresponding months of successive years.
4. Irregular or random movements: These characterize the sporadic motion of time series due to random or chance events, such as labor disputes, floods, or announced personnel changes within companies, political events (such as political events in 2022 have a great impact on market prices)

The trend, cyclic, seasonal, and irregular movements are represented by the variables **T, C, S, I**, respectively. Time-series modeling is also referred to as the decomposition of a time series into these four basic movements. The time-series variable Y can be modeled as either the product of the four variables.

There are two types of motrdels:

Y = T +C+ S+I (additive model)

Y = T x C x S x I (multiplicative model)



Stackoverflow

Data processing technique

Autocorrelation

- Autocorrelation is the correlation between two observations at different points in a time series. When these correlations are present, they indicate that past values influence the current value. Analysts use the autocorrelation and partial autocorrelation functions to understand the properties of time series data, fit the appropriate models, and make forecasts

- Often, one of the first steps in any data analysis is performing regression analysis. However, one of the assumptions of regression analysis is that the data has no autocorrelation. This can be frustrating because if you try to do a regression analysis on data with autocorrelation, then your analysis will be misleading.

Smooth Time Series

Smoothing is the process of removing random variations that appear as coarseness in a plot of raw time series data. It reduces the noise to emphasize the signal that can contain trends and cycles. Analysts also refer to the smoothing process as filtering the data.

Moving average: Moving averages are a series of averages calculated using sequential segments of data points over a series of values. They have a length, which defines the number of data points to include in each average.

Simple Moving Average(SMA) is calculated by taking the unweighted mean of k (size of the window) observations at a time that is present in the current window. It is used for analyzing trends.

$$\frac{y_1+y_2+\cdots+y_n}{n}, \frac{y_2+y_3+\cdots+y_{n+1}}{n}, \frac{y_3+y_4+\cdots+y_{n+2}}{n}, \cdots$$

Example:

Original data	**2**	**3**	**4**	**5**	**9**
Moving average of order 3		(2+3+4)/3=3	(3+4+5)/3=4	6	

Exponential moving average (EMA) tells us the weighted mean of the previous K data points. EMA places a greater weight and significance on the most recent data points. The formula to calculate EMA at the time-period t is:

$$EMAt = at + (1-)EMAt - 1$$

EMAt: Exponential Moving Average at time t
α : degree of decrease in weight of observation with time
At : observation at time t

Time-series forecasting:

There are several models for forecasting:

ARIMA (Auto-Regressive Integrated Moving Average), also known as the Box-Jenkins methodology

2. Similarity Search in Time-Series Analysis

What is a similarity search?

Which find data that match the given query exactly, a similarity search finds data sequences that differ only slightly from the given query sequence

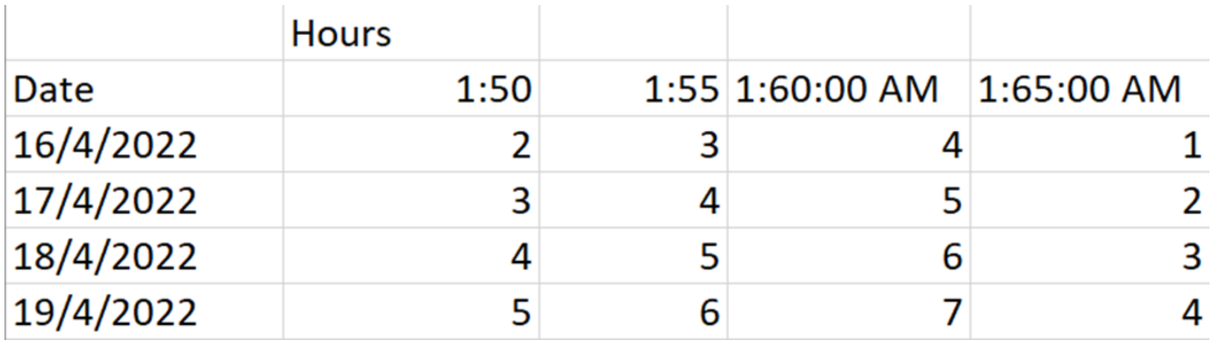
Subsequence matching finds the sequences in S that contain subsequences that are similar to a given query sequence x

Whole sequence matching finds a set of sequences in S that are similar to each other (as a whole).

Application
Similarity search in time-series analysis is useful for financial market analysis (e.g., stock data analysis), medical diagnosis (e.g., cardiogram analysis), and in scientific or engineering databases (e.g., power consumption analysis).

Data Reduction and Transformation Techniques

Due to the tremendous size and high-dimensionality of time-series data, data reduction often serves as the first step in time-series analysis. Data reduction leads to not only much smaller storage space but also much faster processing.
A time-series data table with row as time, column as data. To represent this data table, it is necessary to represent it on a 3-dimensional coordinate axis. So how to reduce the dimension of the data to 2 dimensions for easy visualization ?



	Hours			
Date	1:50	1:55	1:60:00 AM	1:65:00 AM
16/4/2022	2	3	4	1
17/4/2022	3	4	5	2
18/4/2022	4	5	6	3
19/4/2022	5	6	7	4

We propose several dimensionality reduction techniques can be used in time-series analysis. Examples include:

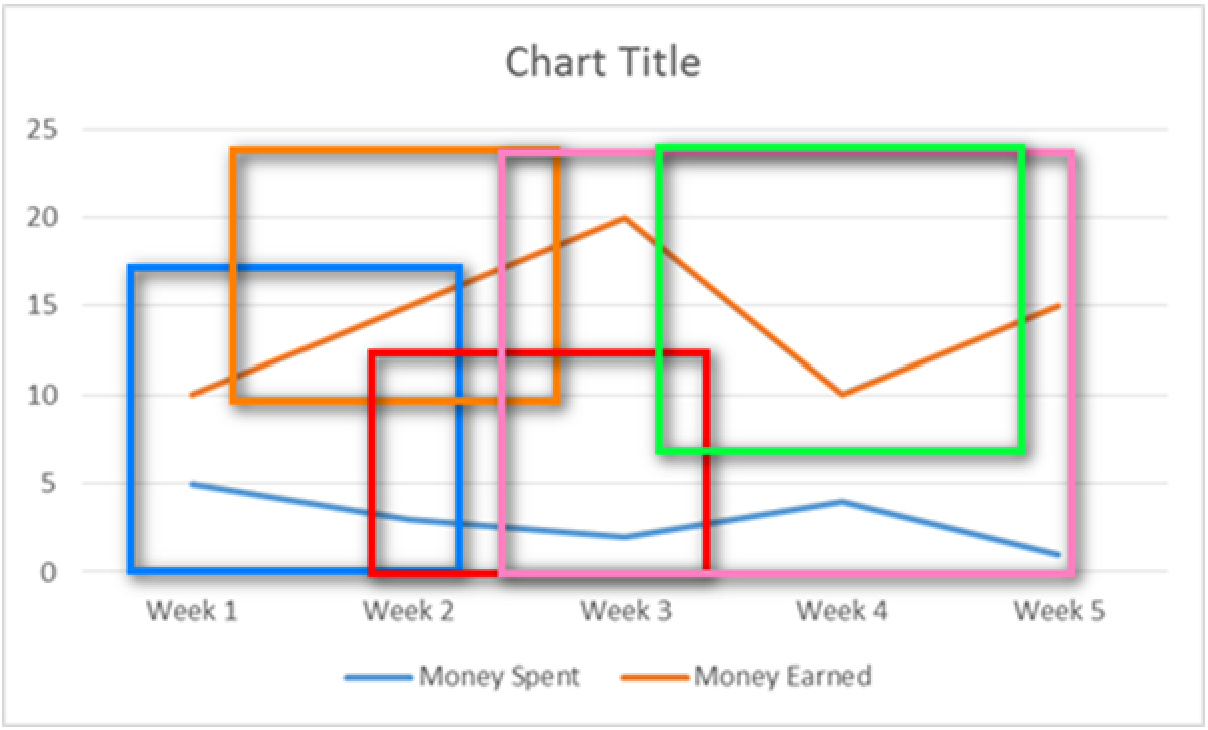
1. The discrete Fourier transform (DFT) as the classical data reduction technique
2. Discrete wavelet transforms (DWT),
3. Singular Value Decomposition (SVD) based on Principle Components Analysis (PCA),
4. Random projection-based sketch techniques

Indexing Methods for Similarity Search

After reducing and transforming, we use the indexing method to access efficiently a multidimensional index

Various kinds of indexing methods have been explored to speed up the similarity search. For example,
R-trees and R*-trees

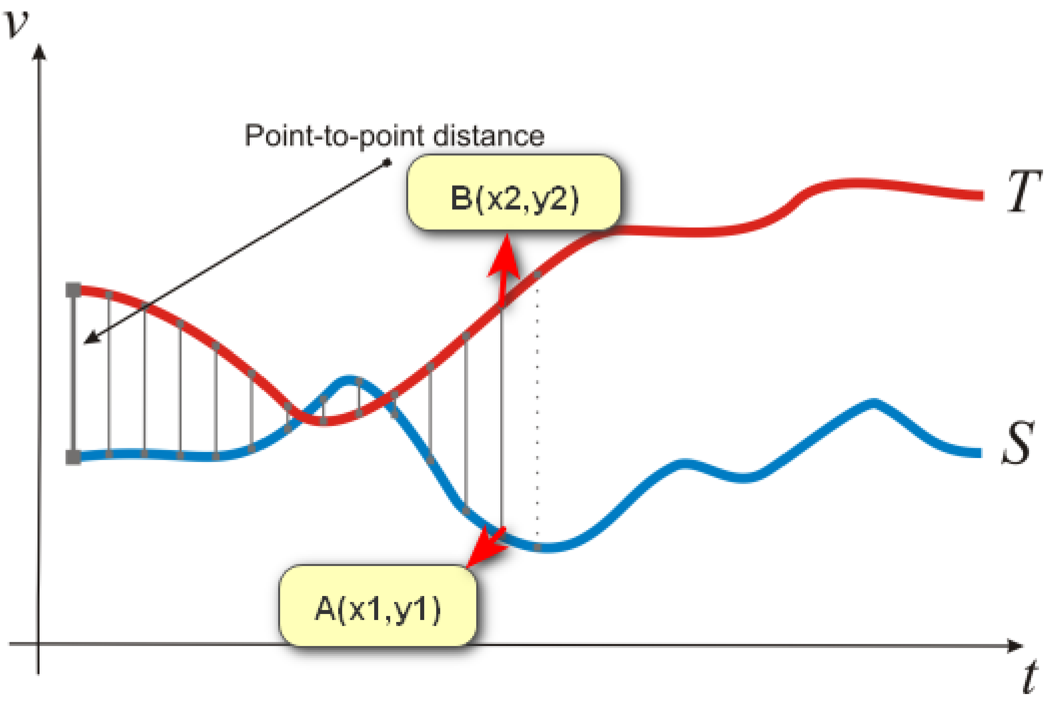
R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates.



divide the chart into many partitions

Similarity Search Methods

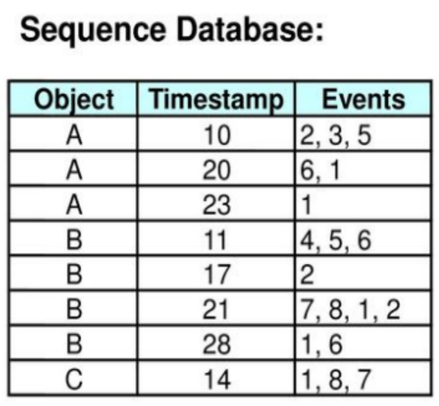
For similarity analysis of time-series data, **Euclidean distance** is typically used as a similarity measure. Here, the smaller the distance between two sets of time-series data, the more similar are the two series



C. Sequence Data

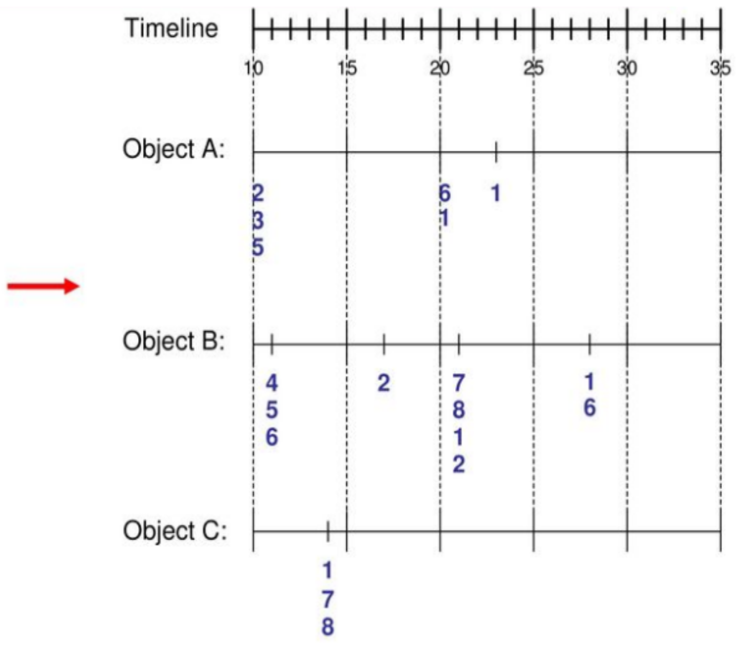
1. Definition of Sequence Data and Sequential Pattern Mining

The sequence of itemsets representing the behavior of a client over a specific period. The database involved in a sequential pattern mining process is a (usually large) set of data sequences.



Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



A sequence is an ordered list of **elements** or **transactions**

s = < e1 e2 e3 … >

Each element is a collection of **events**(**items**), in a specific time.

ei = { i1, i2, i3}

If there is a **k-sequence** mentioned, it is a sequence having k events. In sequence pattern mining, we would find the complete set of **frequent subsequences** from a set of sequences.

2. Sequential Pattern Mining Problem

However, in order to properly mine the sequence database in practical use cases, there are some problems:

- A large potential number of subsequences are hidden inside these sequence dataset.
- This poses a threat to mining sequence data on huge datasets as it relates to memory scalability, work partitioning, and load balancing.
- Older techniques become slower as the databases continuously grow in size. Better techniques and algorithms are required.

3. Methods of Sequential Pattern Mining

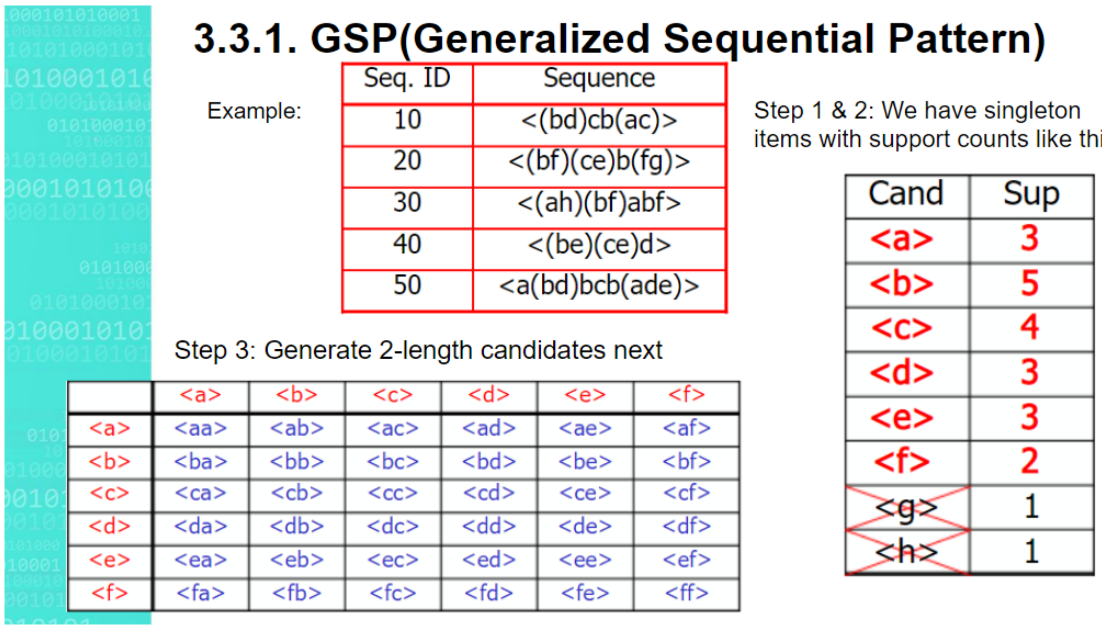
New methods of mining Sequential Pattern must meet the following requirements:

- Find the complete set of frequent subsequence with min_sup
- Highly efficient, scalability, scans
- Be able to incorporate multiple user-constraints.

We will go through three popular techniques for mining Sequential Pattern:

❖ **GSP (Generalized Sequential Pattern)**

- ➢ Generate k-length candidates from the database. Initially, it starts at 1.
- ➢ Scan database to collect support count for each candidate sequence.
- ➢ Repeat step 1 and 2 with (k+1)-length candidates until no candidate can be found.



3.3.1. GSP(Generalized Sequential Pattern)

Example:

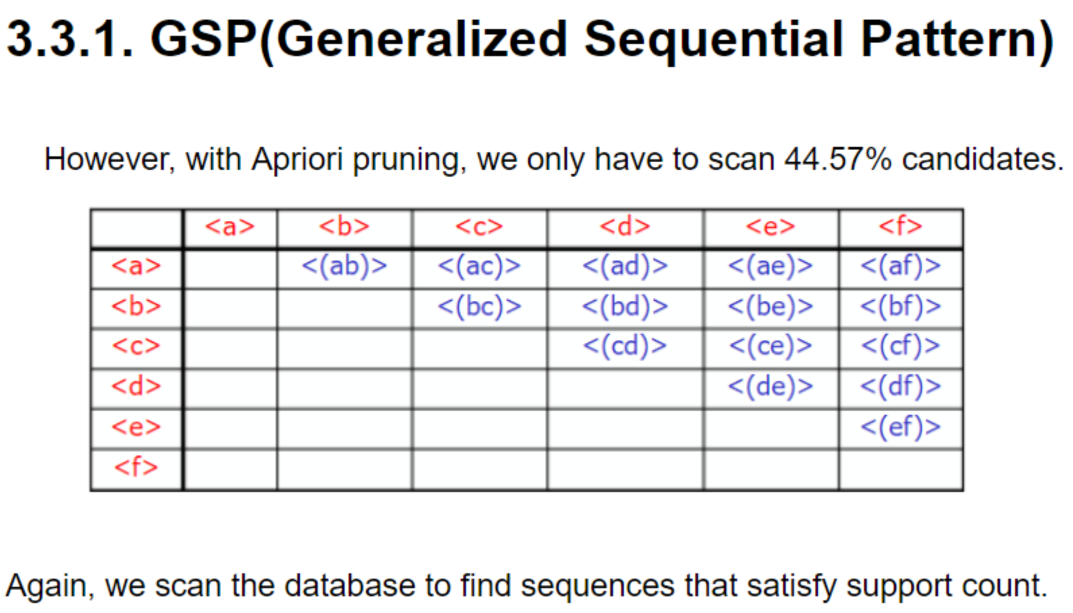
Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Step 1 & 2: We have singleton items with support counts like this

Cand	Sup
<a>	3
<b>	5
<c>	4
<d>	3
<e>	3
<f>	2
~~<g>~~	1
~~<h>~~	1

Step 3: Generate 2-length candidates next

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>



3.3.1. GSP(Generalized Sequential Pattern)

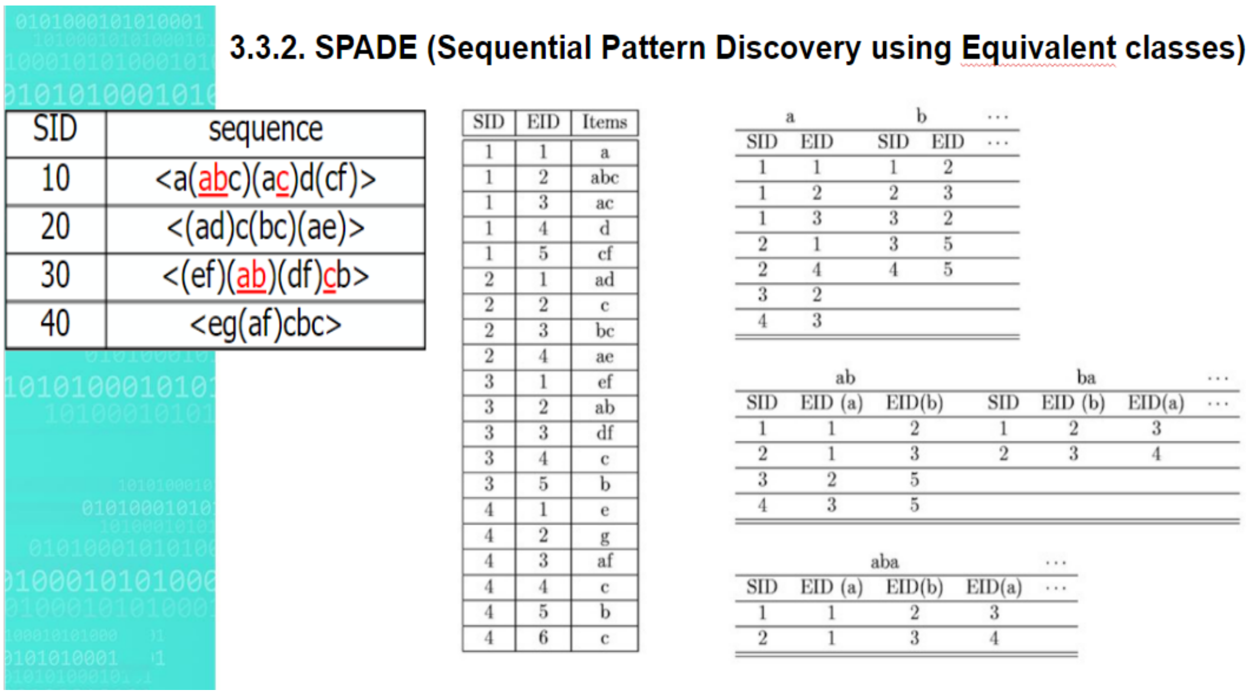
However, with Apriori pruning, we only have to scan 44.57% candidates.

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Again, we scan the database to find sequences that satisfy support count.

Despite the fact that this method still has some problems like generating large sets of subsequence, a number of database scans, it still improves major performance due to reduction of candidates in the second image.

❖ **SPADE (Sequential Pattern Discovery using Equivalent Classes)**
  - ➢ Spade is a vertical format sequential pattern mining method.
  - ➢ Sequence database is mapped to a large set under the format: **<SID,EID>**
  - ➢ Basic steps of Spade are technically similar to GSP, only the format is different.



3.3.2. SPADE (Sequential Pattern Discovery using Equivalent classes)

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

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			

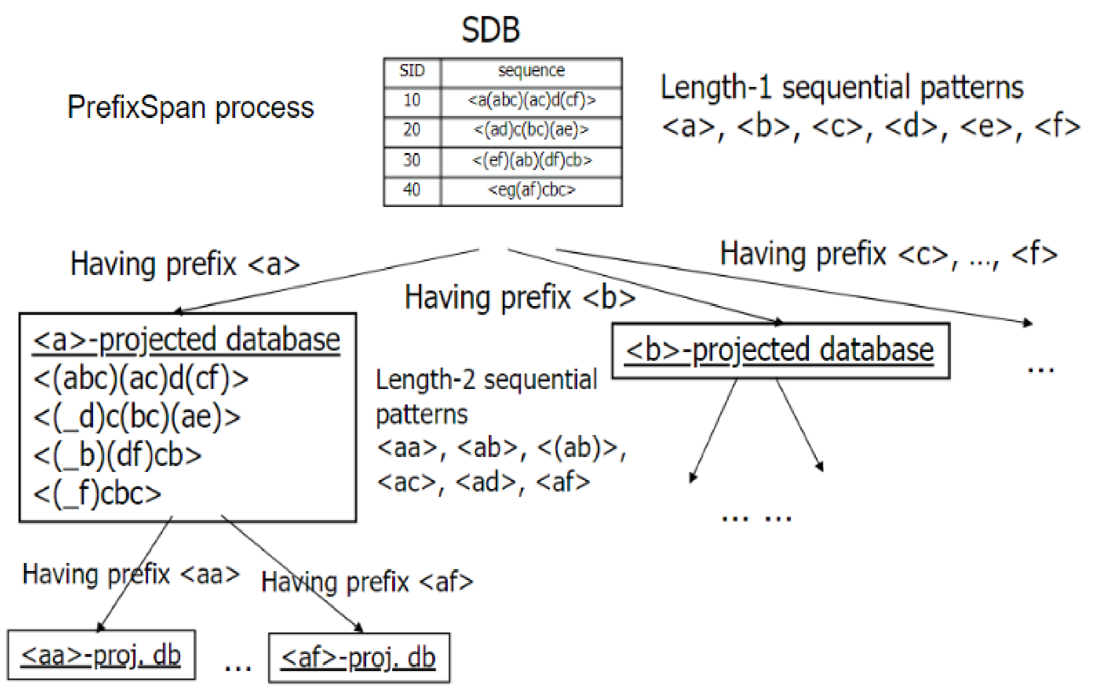
ab			ba			···
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	···
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

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

Again, it still poses some issues just like GSP, the only improvement is that with a different format, it is under better structured data shape and follows **Breadth First Search** to perform mining in small datasets.

❖ **PrefixSpan (Prefix-Projected Sequential Pattern Growth)**
  - ➢ Find k-length sequential patterns(start with k=1)
  - ➢ Divide search space, complete set of sequential patterns are partitioned into 6 subsets.
  - ➢ Generate projected databases from each subset.
  - ➢ Repeat with (k+1)-length sequential patterns until no candidates are found.

3.3.3. PrefixSpan (Prefix-Projected Sequential Pattern Growth)



Better than both mining methods above as no candidate sequence needs generating, projected databases keep shrinking, However, in order to give better results, the databases are required to be projected which can be improved by pseudo-projections. Referred to this document for more information:
https://www.semanticscholar.org/paper/Prefix-Span-Algorithm-with-Pseudo-Projection-for-2Thirukumar/6857c3f2191a448e06673d27a1de5acf9829e384

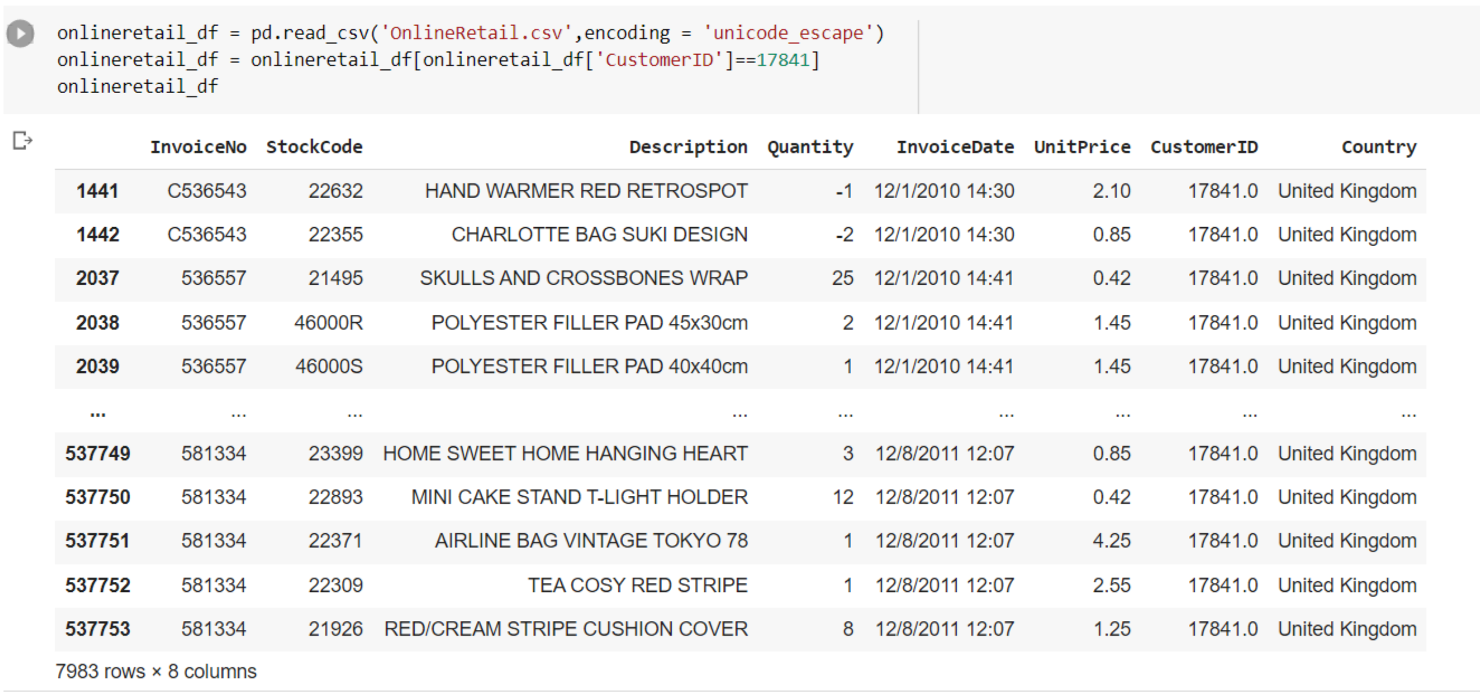
4. Constraints based mining of Sequential Pattern

Constraint-based mining incorporates user-specified constraints to reduce the search space and derive only patterns that are of interest to the user.

Constraints can be expressed in many forms:

- **Anti-monotonic**: maximal length defined. Given a constraint T<10, if a sequence does not satisfy then neither of its supersequences
- **Motonic**: minimum length defined. Given a constraint T>10, that a sequence satisfies the condition, all of its supersequence are expected to meet requirements too..
- **Succinct**: relates to a specific duration. Given a constraint year = 2005, we can filter whose year 2005 before mining which leads to efficiency improvement.

III. Solved Problem



```
onlineretail_df = pd.read_csv('OnlineRetail.csv',encoding = 'unicode_escape')
onlineretail_df = onlineretail_df[onlineretail_df['CustomerID']==17841]
onlineretail_df
```

```
        InvoiceNo  StockCode                      Description  Quantity     InvoiceDate  UnitPrice  CustomerID         Country
1441      C536543      22632          HAND WARMER RED RETROSPOT        -1  12/1/2010 14:30       2.10     17841.0  United Kingdom
1442      C536543      22355           CHARLOTTE BAG SUKI DESIGN       -2  12/1/2010 14:30       0.85     17841.0  United Kingdom
2037       536557      21495        SKULLS AND CROSSBONES WRAP        25  12/1/2010 14:41       0.42     17841.0  United Kingdom
2038       536557     46000R        POLYESTER FILLER PAD 45x30cm        2  12/1/2010 14:41       1.45     17841.0  United Kingdom
2039       536557     46000S        POLYESTER FILLER PAD 40x40cm        1  12/1/2010 14:41       1.45     17841.0  United Kingdom
...           ...        ...                              ...       ...              ...        ...         ...             ...
537749     581334      23399     HOME SWEET HOME HANGING HEART         3  12/8/2011 12:07       0.85     17841.0  United Kingdom
537750     581334      22893        MINI CAKE STAND T-LIGHT HOLDER    12  12/8/2011 12:07       0.42     17841.0  United Kingdom
537751     581334      22371         AIRLINE BAG VINTAGE TOKYO 78      1  12/8/2011 12:07       4.25     17841.0  United Kingdom
537752     581334      22309                   TEA COSY RED STRIPE     1  12/8/2011 12:07       2.55     17841.0  United Kingdom
537753     581334      21926      RED/CREAM STRIPE CUSHION COVER       8  12/8/2011 12:07       1.25     17841.0  United Kingdom

7983 rows × 8 columns
```

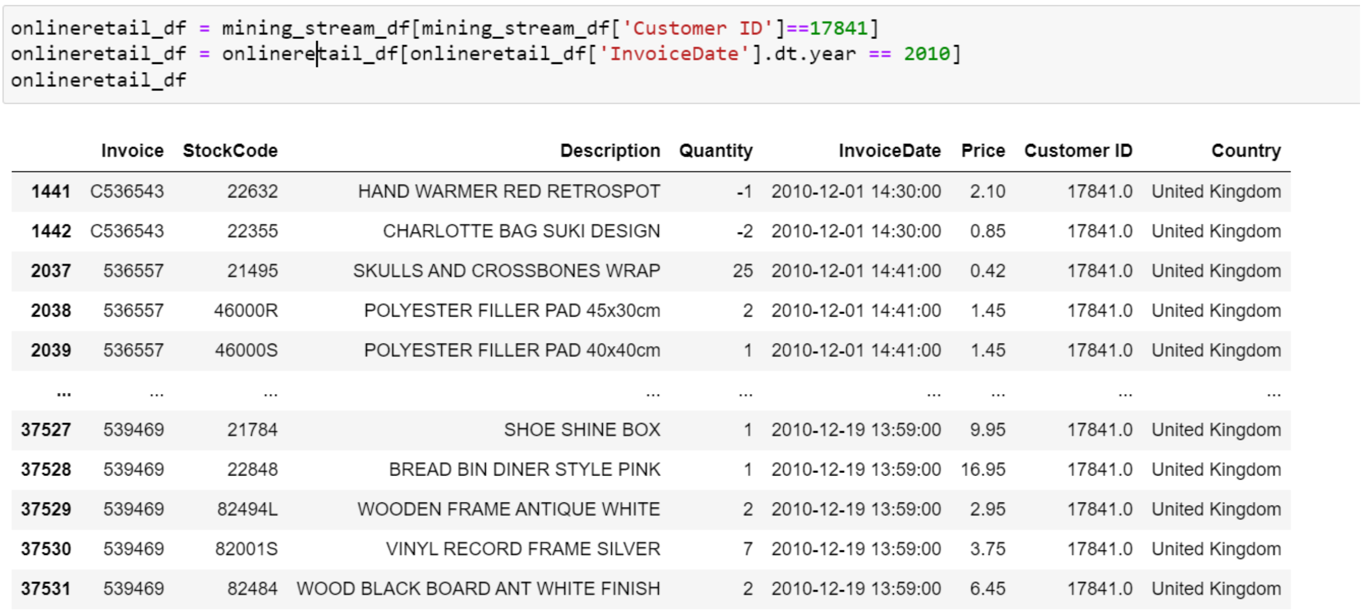
Đây là một tập dữ liệu về các tất cả các giao dịch đã xảy ra trong thị trường mua bán Anh bắt đầu từ năm 1/12/2010 đến 9/12/2011 với các biến:

1. **InvoiceNo**: Invoice number. Nominal. A 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.
2. **StockCode**: Product (item) code. Nominal. A 5-digit integral number uniquely assigned to each distinct product.
3. **Description**: Product (item) name. Nominal.
4. **Quantity**: The quantities of each product (item) per transaction. Numeric.
5. **InvoiceDate**: Invoice date and time. Numeric. The day and time when a transaction was generated.
6. **UnitPrice**: Unit price. Numeric. Product price per unit in sterling (Â£).
7. **CustomerID**: Customer number. Nominal. A 5-digit integral number uniquely assigned to each customer.
8. **Country**: Country name. Nominal. The name of the country where a customer resides.

Nhóm sẽ giới hạn lại bằng cách khai thác và phân tích hành vi mua hàng của **một khách hàng** nhất định để cho kết quả tốt nhất. Khách hàng được chọn ở đây có **CustomerID = 1784** bởi vì đây là khách hàng mua nhiều hàng hóa nhất trên toàn bộ tập dữ liệu với gần 8k record được ghi.

B. Mining Stream

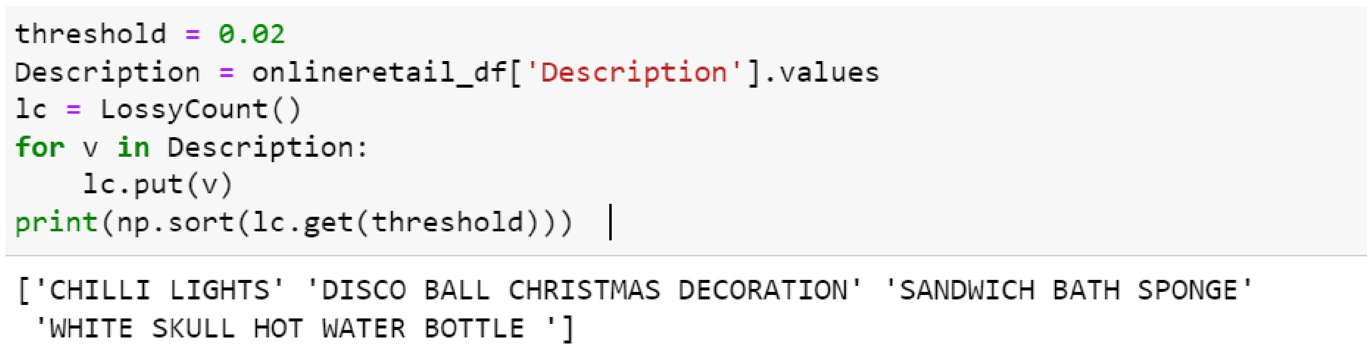
Ví dụ về thuật toán Lossy Counting.



```
onlineretail_df = mining_stream_df[mining_stream_df['Customer ID']==17841]
onlineretail_df = onlineretail_df[onlineretail_df['InvoiceDate'].dt.year == 2010]
onlineretail_df
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
1441	C536543	22632	HAND WARMER RED RETROSPOT	-1	2010-12-01 14:30:00	2.10	17841.0	United Kingdom
1442	C536543	22355	CHARLOTTE BAG SUKI DESIGN	-2	2010-12-01 14:30:00	0.85	17841.0	United Kingdom
2037	536557	21495	SKULLS AND CROSSBONES WRAP	25	2010-12-01 14:41:00	0.42	17841.0	United Kingdom
2038	536557	46000R	POLYESTER FILLER PAD 45x30cm	2	2010-12-01 14:41:00	1.45	17841.0	United Kingdom
2039	536557	46000S	POLYESTER FILLER PAD 40x40cm	1	2010-12-01 14:41:00	1.45	17841.0	United Kingdom
...	...	...	...	...	...	...	...	...
37527	539469	21784	SHOE SHINE BOX	1	2010-12-19 13:59:00	9.95	17841.0	United Kingdom
37528	539469	22848	BREAD BIN DINER STYLE PINK	1	2010-12-19 13:59:00	16.95	17841.0	United Kingdom
37529	539469	82494L	WOODEN FRAME ANTIQUE WHITE	2	2010-12-19 13:59:00	2.95	17841.0	United Kingdom
37530	539469	82001S	VINYL RECORD FRAME SILVER	7	2010-12-19 13:59:00	3.75	17841.0	United Kingdom
37531	539469	82484	WOOD BLACK BOARD ANT WHITE FINISH	2	2010-12-19 13:59:00	6.45	17841.0	United Kingdom

Ta áp dụng thuật toán Lossy Counting vào với threshold = 0.02, tức là ta quan tâm tới những mặt mà khách hàng này đã mua lớn hơn 2% tổng số hàng đã mua



```
threshold = 0.02
Description = onlineretail_df['Description'].values
lc = LossyCount()
for v in Description:
    lc.put(v)
print(np.sort(lc.get(threshold)))
```

```
['CHILLI LIGHTS' 'DISCO BALL CHRISTMAS DECORATION' 'SANDWICH BATH SPONGE'
 'WHITE SKULL HOT WATER BOTTLE ']
```

Nhận xét & Đánh giá:

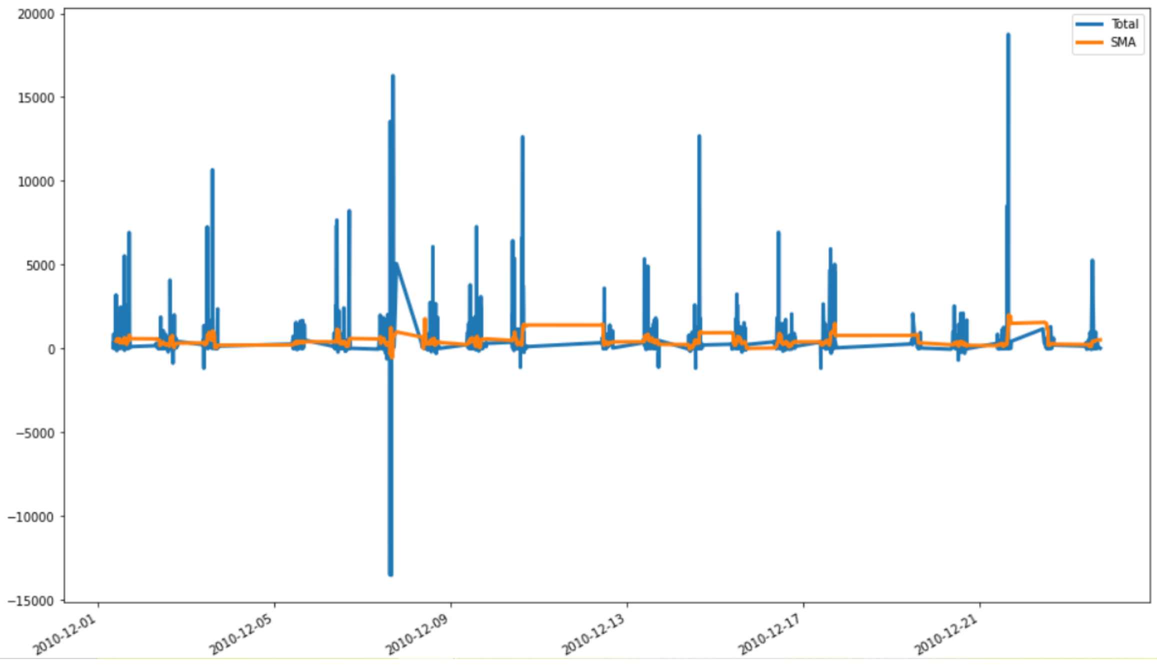
Qua thuật toán Lossy Counting, cùng với threshold ta biết được tỉ lệ những sản phẩm mà khách hàng đã mua, áp dụng trên data lớn ta sẽ tìm ra được những mặt hàng nào bán chạy, mặt hàng nào đang ế.

Từ đó mà ta có những hành vi điều chỉnh thích hợp như tăng, giảm giá sản phẩm phù hợp, nhập thêm vào những mặt hàng đang bán chạy để đáp ứng nhu cầu người dùng.

C. Time-Series

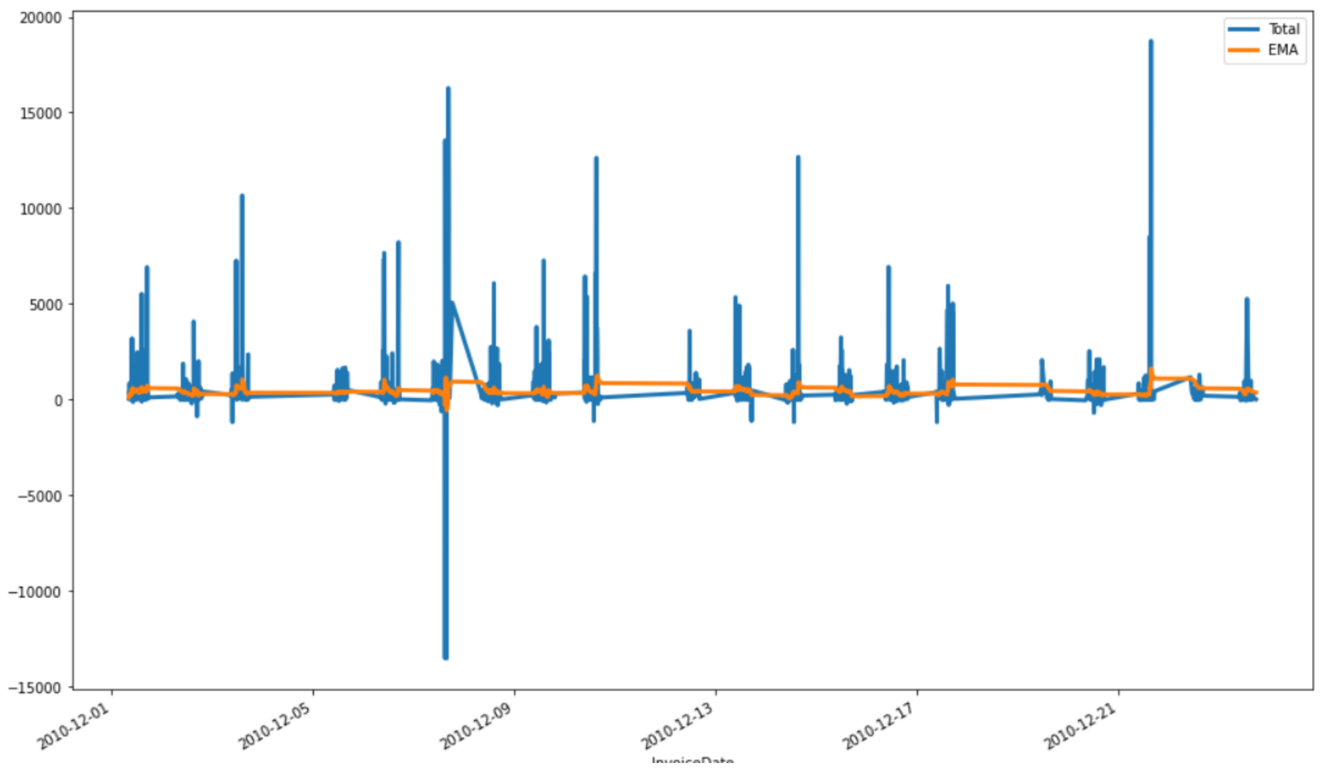
Ví dụ về cách làm trơn dữ liệu bằng phương pháp Simple Moving Average. Với đường màu xanh là dữ liệu gốc. Đường màu cam là dữ liệu sau khi dung **SMA**.

```
In [15]: time_series_df['SMA'] = time_series_df['Total'].rolling(20).mean()
         time_series_df[['Total','SMA']].plot(label='RELIANCE',figsize=(16, 10),linewidth=3)

Out[15]: <AxesSubplot:xlabel='InvoiceDate'>
```



Ví dụ về cách làm trơn dữ liệu bằng phương pháp **Exponential moving average (EMA) .** Với đường màu xanh là dữ liệu gốc. Đường màu cam là dữ liệu sau khi dung **EMA**

```
time_series_df['EMA'] = time_series_df['Total'].ewm(20).mean()
time_series_df[['Total','EMA']].plot(label='RELIANCE',figsize=(16, 10),linewidth=3)

<AxesSubplot:xlabel='InvoiceDate'>
```

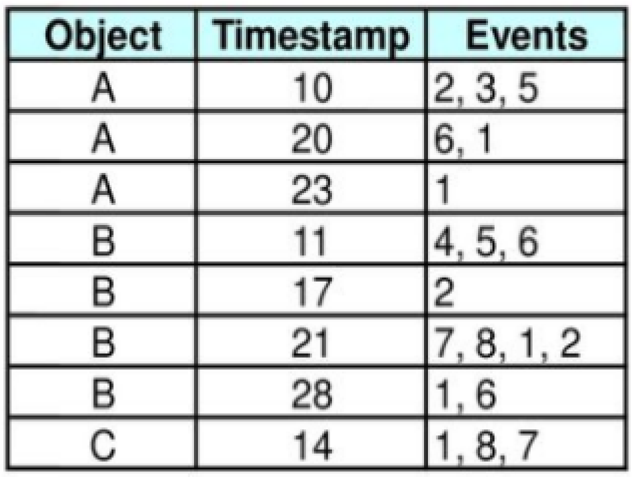


Nhận xét: Với đường dữ liệu gốc trong dữ liệu chuỗi thời gian thường biến động rất nhanh (lên hoặc xuống) từ đó khi trực quan không thể nào quan sát được xu hướng của dữ liệu. Do đó chúng ta phải dùng phương pháp làm trơn dữ liệu để thấy rõ xu hướng. Về hai phương pháp này thì **EMA** làm trơn dữ liệu tốt hơn **SMA** vì **EMA** có độ phức tạp cao hơn

D. Sequence Data

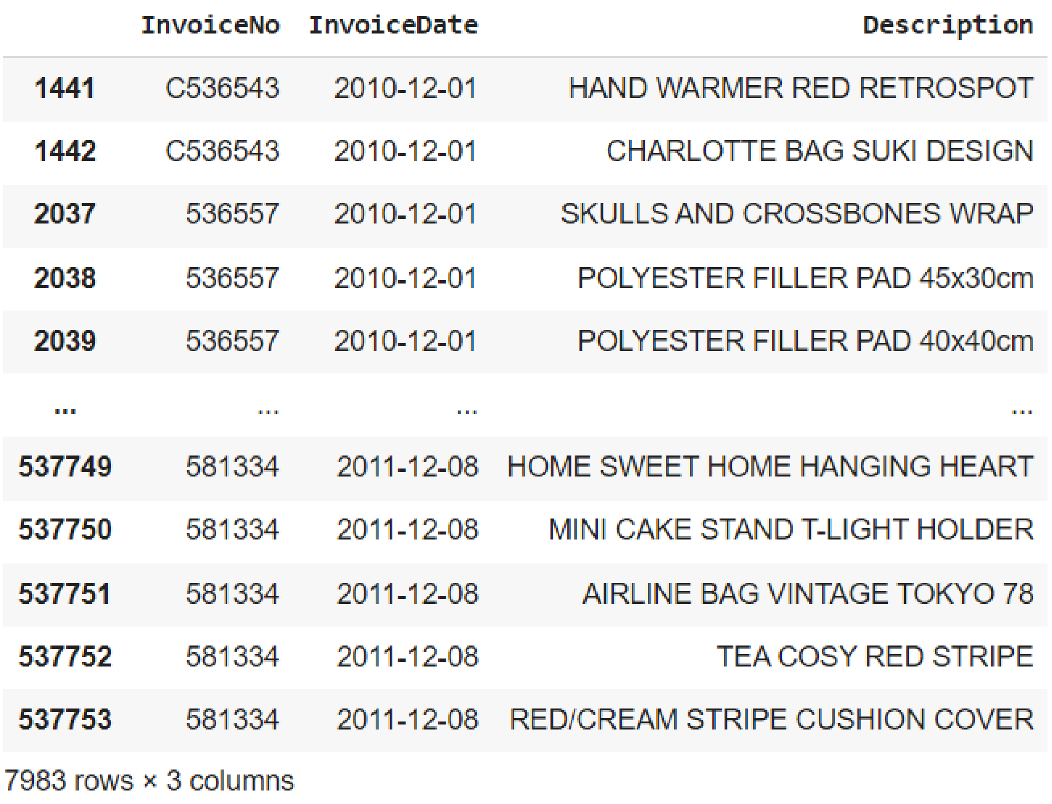
Ở phần 3 Sequence data, nhóm xin giới thiệu thuật toán khai thác PrefixSpan bằng Spark.

Mục đích cuối cùng cho việc phân tích chính là ta đạt được một sequence database có dạng như sau:



Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7

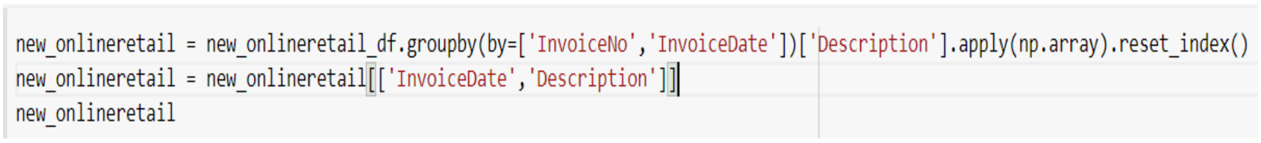
Tiếp theo, nhóm thực hiện tiền xử lý trên tập dữ liệu. Đầu tiên ta sẽ đổi cột InvoiceDate thành kiểu datetime và bỏ thời gian (hour:minute). Lý do cho việc này bởi vì từng hóa đơn chỉ được ghi trong chính xác một thời gian nhất định. Thí dụ nếu một người mua vào ngày 1/12/2010 vào lúc 14:30 sẽ được tag là một hóa đơn có ID riêng. Nếu vào 14:31 cùng ngày, người đó mua thêm thì hóa đơn sẽ được tag một ID khác và các dòng này đã được sắp xếp theo thời gian. Như vậy, ta sẽ không cần chính xác thời gian trong ngày.



	InvoiceNo	InvoiceDate	Description
1441	C536543	2010-12-01	HAND WARMER RED RETROSPOT
1442	C536543	2010-12-01	CHARLOTTE BAG SUKI DESIGN
2037	536557	2010-12-01	SKULLS AND CROSSBONES WRAP
2038	536557	2010-12-01	POLYESTER FILLER PAD 45x30cm
2039	536557	2010-12-01	POLYESTER FILLER PAD 40x40cm
...	...	...	...
537749	581334	2011-12-08	HOME SWEET HOME HANGING HEART
537750	581334	2011-12-08	MINI CAKE STAND T-LIGHT HOLDER
537751	581334	2011-12-08	AIRLINE BAG VINTAGE TOKYO 78
537752	581334	2011-12-08	TEA COSY RED STRIPE
537753	581334	2011-12-08	RED/CREAM STRIPE CUSHION COVER

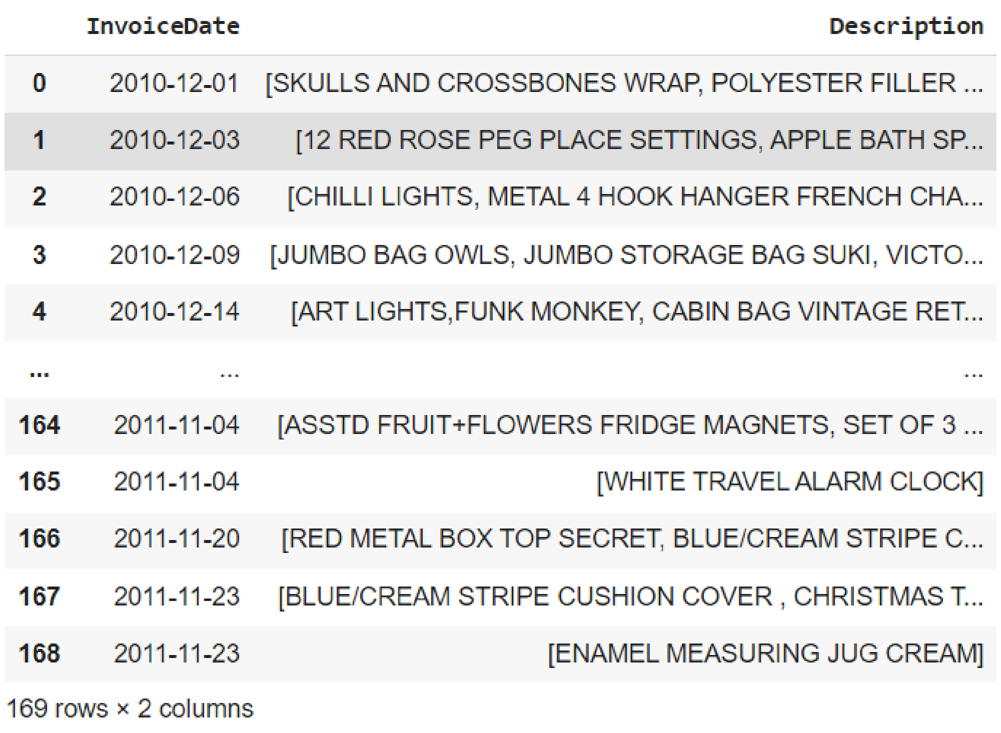
7983 rows × 3 columns

Bởi vì nhóm sử dụng Spark để thực hiện thuật toán khai thác PrefixSpan, dataframe cần thiết sẽ nằm trong các numpy array được cấu hình sẵn, ta sẽ cần biến đổi một chút. Sử dụng groupby lên InvoiceNo và InvoiceDate với các dữ kiện ở cột description được nối với nhau tạo thành một numpy array.



```
new_onlineretail = new_onlineretail_df.groupby(by=['InvoiceNo','InvoiceDate'])['Description'].apply(np.array).reset_index()
new_onlineretail = new_onlineretail[['InvoiceDate','Description']]
new_onlineretail
```

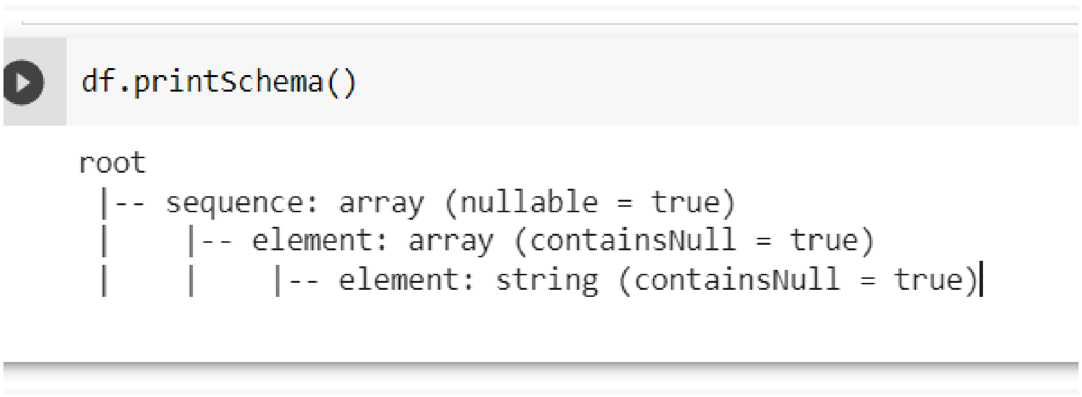
Một điều nữa là tới lúc này ta không cần Invoice No nữa bởi vì ta đã gộp các hóa đơn trong cùng một ngày thành một numpy array ở cột description. Cuối cùng, ta đạt được một sequence database như sau:



	InvoiceDate	Description
0	2010-12-01	[SKULLS AND CROSSBONES WRAP, POLYESTER FILLER ...
1	2010-12-03	[12 RED ROSE PEG PLACE SETTINGS, APPLE BATH SP...
2	2010-12-06	[CHILLI LIGHTS, METAL 4 HOOK HANGER FRENCH CHA...
3	2010-12-09	[JUMBO BAG OWLS, JUMBO STORAGE BAG SUKI, VICTO...
4	2010-12-14	[ART LIGHTS,FUNK MONKEY, CABIN BAG VINTAGE RET...
...	...	...
164	2011-11-04	[ASSTD FRUIT+FLOWERS FRIDGE MAGNETS, SET OF 3 ...
165	2011-11-04	[WHITE TRAVEL ALARM CLOCK]
166	2011-11-20	[RED METAL BOX TOP SECRET, BLUE/CREAM STRIPE C...
167	2011-11-23	[BLUE/CREAM STRIPE CUSHION COVER , CHRISTMAS T...
168	2011-11-23	[ENAMEL MEASURING JUG CREAM]

169 rows × 2 columns

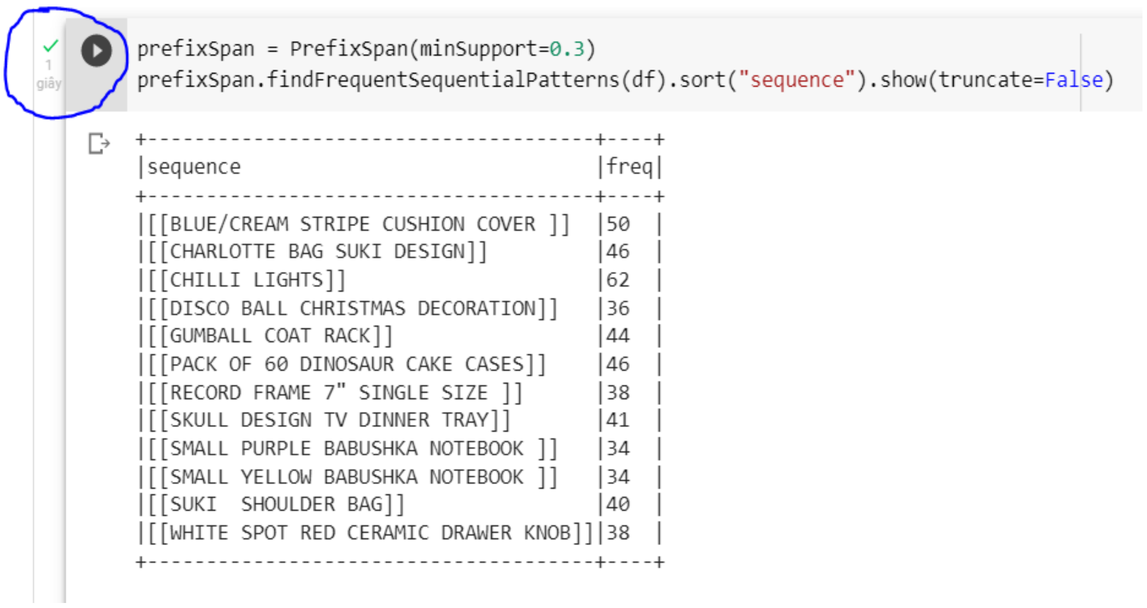
Từ gần 8k dòng chỉ còn 169 dòng có nghĩa ta gộp các record thành một chuỗi ở cột Description dựa theo hóa đơn thành công. Tức là mỗi dòng ở cột Description chính là một event(item). Việc cuối cùng ta cần làm là gộp các item trong cùng một ngày để tạo thành một sequence, tức là mỗi ngày ta sẽ được một sequence, nhiều ngày sẽ tạo thành một sequence database hoàn chỉnh cho việc khai thác dữ liệu. Ta được sequence database có schema cuối cùng như sau:



```
df.printSchema()
```

```
root
 |-- sequence: array (nullable = true)
 |    |-- element: array (containsNull = true)
 |    |    |-- element: string (containsNull = true)
```

Sai khi áp dụng PrefixSpan trên dataframe, ta được kết quả khai thác:



```
prefixSpan = PrefixSpan(minSupport=0.3)
prefixSpan.findFrequentSequentialPatterns(df).sort("sequence").show(truncate=False)
```

```
+-------------------------------------+----+
|sequence                             |freq|
+-------------------------------------+----+
|[[BLUE/CREAM STRIPE CUSHION COVER ]] |50  |
|[[CHARLOTTE BAG SUKI DESIGN]]        |46  |
|[[CHILLI LIGHTS]]                    |62  |
|[[DISCO BALL CHRISTMAS DECORATION]]  |36  |
|[[GUMBALL COAT RACK]]                |44  |
|[[PACK OF 60 DINOSAUR CAKE CASES]]   |46  |
|[[RECORD FRAME 7" SINGLE SIZE ]]     |38  |
|[[SKULL DESIGN TV DINNER TRAY]]      |41  |
|[[SMALL PURPLE BABUSHKA NOTEBOOK ]]  |34  |
|[[SMALL YELLOW BABUSHKA NOTEBOOK ]]  |34  |
|[[SUKI  SHOULDER BAG]]               |40  |
|[[WHITE SPOT RED CERAMIC DRAWER KNOB]]|38  |
+-------------------------------------+----+
```

Đánh giá kết quả:

Như vậy để khai thác 8000 dòng dữ liệu từ một khách hàng sau khi biến đổi thành một sequence database hoàn chỉnh, thuật toán chỉ cần 1s để tìm ra được frequent subsequences.

Với minSupport = 0.3, ta đạt được các kết quả rất tốt như BLUE/CREAM STRIPE CUSHION COVER - 50 lần, CHILLI LIGHTS - 62 lần, PACK OF 60 DINOSAUR CAKE CASES - 46 lần, etc.

Bằng cách khai thác như vậy, ta có thể đưa ra các mặt hàng cũng như hành vi mua hàng thường diễn ra của một người. Hơn nữa, ta có thể dự đoán hành vi mua hàng trong các khoảng thời gian tiếp theo của người tiêu dùng này và thực hiện phân phối, điều chỉnh giá trong các sản phẩm liên quan để thu hút cũng như hướng tới nhiều người tiêu dùng hơn.

IV. Self-assessment of the team's work results

Nhìn chung, nhóm đã tận lực nghiên cứu chi tiết các vấn đề của đề tài được cho cũng như phân tích, đánh giá, so sánh các cách khác nhau trong từng vấn đề. Bên cạnh đó, nhóm cũng sử dụng một tập dữ liệu thực tế để thực hiện khai thác bằng các thuật toán, phương pháp nhóm đã giới thiệu.

V. Link google colab

https://colab.research.google.com/drive/12BbsKR-AOatqM6YQza5D4L_mt5RPvwf3?usp=sharing

VI. Link video demo

https://drive.google.com/drive/folders/1iVyLjXal7OnoZS8xadfXDDbZIiB50Jh7?usp=sharing

VII. Reference

Book: Data Mining:Concepts and Techniques Second Edition

https://spark.apache.org/docs/latest/ml-frequent-pattern-mining.html

https://en.wikipedia.org/wiki/Sequence_database

R*-tree - Wikipedia

influxdata.com

How to Calculate Moving Averages in Python? - GeeksforGeeks

https://en.wikipedia.org/wiki/Lossy_Count_Algorithm

https://stackoverflow.com/questions/8033012/what-is-lossy-counting

https://gist.github.com/giwa/bce63f3e2bd493167d92