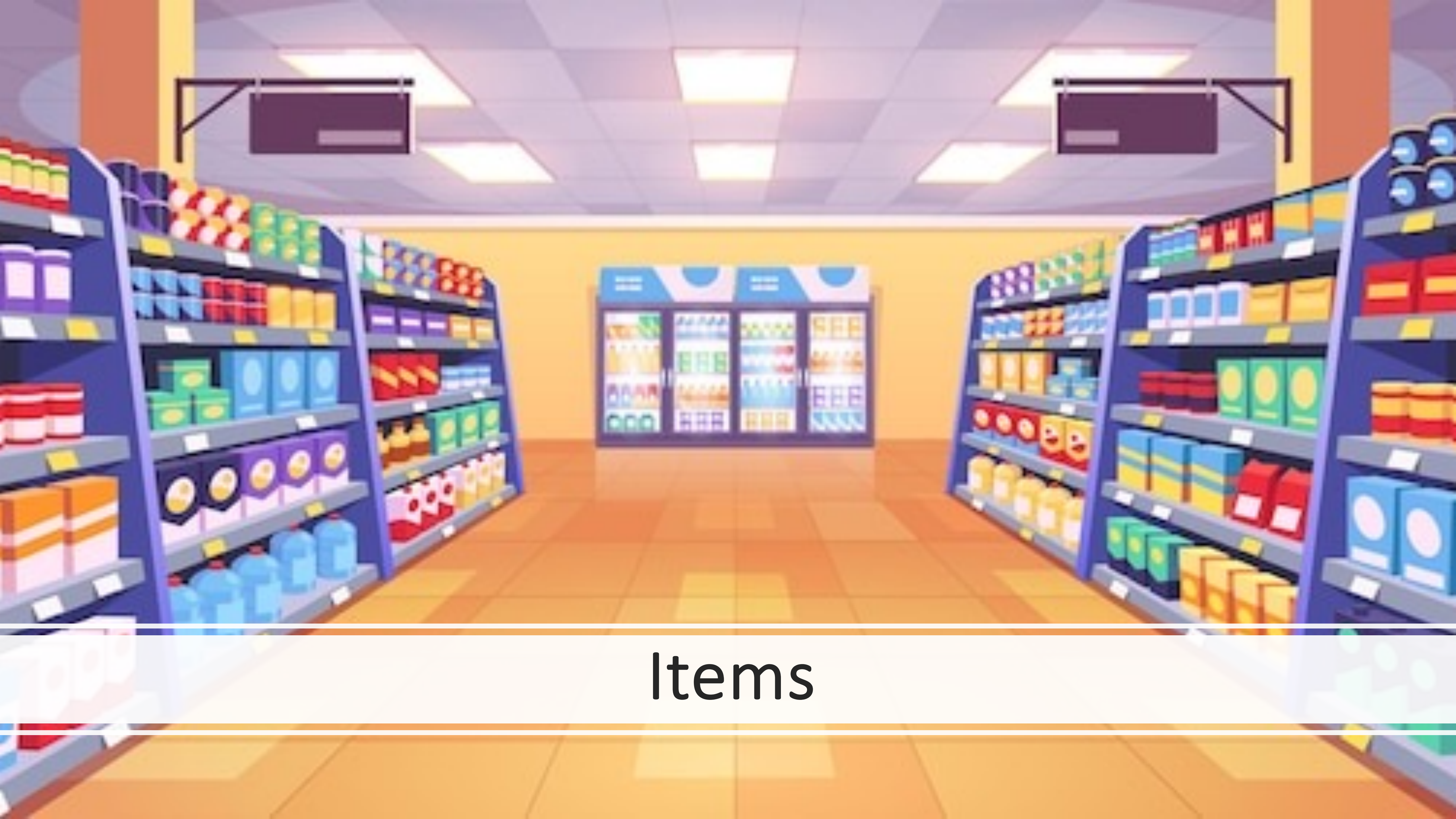
Frequent Itemset Mining

- Ravi Kumar Gupta
- https://kravigupta.in

Mining

- Mining – Process of extracting something valuable
- Data Mining – Process of discovering patterns, correlations, anomalies, relationships within the data

- Consider a supermarket with a lot of products
- People buy stuff.
- In the bill – all products are **items**
- Sets of those products - **itemset**



Items

Itemsets

CASH RECEIPT
Dolore sit amet 97b
«DOLORE»
ID: **684jg77
CODE **9786UJY6643
eiusmod 4.07
adipiscing elit 75.0
ullamco 93.09
voluptate 8.06
TOTAL: 180.22
TAX: 14% 25.02
CHANGE 57.03
INN **454759900
A.B.C «DOLORE»
Lorem ipsum 37a, sit amet 3
www.dolore
(078)***6733 01
ID **#4678J code #8805
CASH RECEIPT
YOUR ORDER NUMBER
★025★
velit esse 40.8
proident 7.07
dolor sit 53.09
deserunt 32.01
minim veniam 9.76
aute irure 6.9
reprehenderit 8.09
TOTAL 157.72
DD/MM/YYYY
RECEIPT
Sit Amet
HOT LINE: 00066742222
ID #0064336
0001 01
CASH#1 #
JPI0744 #
TAX [20%] 30.5
esse cillum 21.35
non proident 4.58
commodo 7.30
veniam 4.06
pariatur 4.09
amet 3.06
magna 9.7
ipsum 40.5
cupidatat 35.20
THANK YOU
FOR YOUR
PURCHASE!
STORE
Dolor sit amet
ID: #H9798978
irure dolor 09.6
Ut enim 35.09
consequat 6.8
nostrud 44.07
Duis 8.0
minim veniam 33.00
dolor sit 6.9
INN 990897**65 09:08:12
DD/MM/YYYY
Online
57747 Quis Nostrud
Officia №1
ID #5790949Y
CODE *UI7YJS66
U.O.S 00968544
consectetur 4.9
dolore 8.09
mollit 55.8
sunt in culp 34.7
duis aute 44.03
minim veniam 20.4
sint 7.98
TAX
15% 26.07$
PAY
CASH 580.4$
Check number #ghy577
30.09.21 13:03:21
FN 0035 Online
ID : UUPF**JN#K900
CASH ≡ 967.90
Duis aute irure ≡ 0.00
APPROVED
Laboris nisi 0079673
Excepteur sint 084278940800
CASHIER: John Smith
69#UKO**KYJF779-8795433
TOTAL ≡ 967.90
CONSECTETUR [10%] 96.7
IPSUM [15%] 145.05
THANK YOU!
CASH
Proident 6.98
Irure dolor 34.90
Laborum 22.65
Excepteur sint 68.09
Duis aute 44.06
Lorem 33.8
Ut enim ad 58.2
12/18/01 15:34:03
code : #7uu6677if
AMET
Dolore eu fugiat nulla
LOREM IPSUM
Ut enim ad
0074578844849000
DATE 21.08/2021
SENDER
Dolor sit amet
Excepteur Sint Occaecat
Anim id est laborum #28
tel. 08778463
DESTINATION
Quis nostrud exercitation
Enim Ad Veniam
tel. 67906796
Duis aute: laboris nisi
WEIGHT 0.5kg
COST 80$
Laborum Officia

Recall the Example

Fathers

Who are sent to buy diapers

Might pickup beers for themselves.

Frequent Itemset Mining

- Goal: Find products frequently bought together.

Frequent itemsets

- Items which are frequently purchased together
- Meets a "minimum support" threshold in database.

- **Support**:
- How many times an itemset is in transactions
- Example: 10 out of 100 transactions show milk and bread together
  - → Support = 0.1 for {milk, bread}.

Overview

Role in Data Mining:

- Key foundation for numerous data mining tasks.
- Aids in detecting patterns such as association rules, correlations, sequences, and more.

Application:

- Popularly used for discovering association rules.
- Identifies sets of items or characteristics frequently co-occurring in databases.

Market-Basket Model

Market-Basket Model

Basket Composition:

- Contains a set of items known as an itemset.
- Number of items in a basket usually small compared to total items.
- Total baskets typically very large, often beyond main memory capacity.

Items and Baskets:

- Items aren't strictly "contained" in baskets.
- Focus on co-occurrences of items in relation to a basket.
- General definition:
  - I={i1,...,ik} : Set of k items.
  - B={b1,...,bn} : Set of n item subsets. Each bi is a basket.

Market-Basket Model

- **Examples**:
- **Retail**:
  - I represents items in a store.
  - Each basket is a purchase transaction.
- **Document Basket**:
  - I includes dictionary words and proper nouns.
  - Each basket is a single document containing words from the document.

Frequent-Itemset Mining

- **Definitions**:
  - I={i1,…,ik}: Set of items.
  - D: Task-relevant data consisting of database transactions.
  - Each transaction T: A subset of items from I such that T⊆I.
  - Every transaction has an identifier: TID – transaction id
  - A transaction T contains set A if and only if A⊆T.
- **Itemset**:
  - Collection of items.
  - An itemset with k distinct items is termed a k-itemset.
  - Example: {computer, anti-virus software, printer, flash-drive} is a 4-itemset.
  - {Bread, butter, Milk} is a 3-itemset

Frequent-Itemset Mining

- **Occurrence Frequency**:
  - Number of transactions that have the itemset.
  - Also called **frequency**, **support count**, or **itemset count**.
- **Frequent Itemset**:
  - An itemset is "frequent" if its support count meets a certain threshold.
  - A minimum support s is defined.
  - An itemset I with support ≥ s is deemed a frequent itemset.

Example

- **Items = {milk (m), coke (c), pepsi (p), beer (b), juice (j)}**
- **Minimum support *s* = 3**

Transactions

1. **T1 = {m, c, b}**
2. **T2 = {m, p, j}**
3. **T3 = {m, b}**
4. **T4 = {c, j}**
5. **T5 = {m, p, b}**
6. **T6 = {m, c, b, j}**
7. **T7 = {c, b, j}**
8. **T8 = {b, c}**

Find the Frequent Itemsets?

Transactions

- **T1 = {m, c, b}**
- **T2 = {m, p, j}**
- **T3 = {m, b}**
- **T4 = {c, j}**
- **T5 = {m, p, b}**
- **T6 = {m, c, b, j}**
- **T7 = {c, b, j}**
- **T8 = {b, c}**

Count support for individual item

Item	Count
m	5 (T1, T2, T3, T5, T6)
c	5 (T1, T4, T6, T7, T8)
b	6 (T1, T3, T5, T6, T7, T8)
p	2 (T2, T5)
j	4 (T2, T4, T6, T7)

{m}, {c}, {b}, {j}

Count the support for 2-itemsets

Itemset	Count	Transactions
{m, c}	3	T1, T6
{m, b}	4	T1, T3, T5, T6
{m, j}	2	T2, T6
{c, b}	5	T1, T6, T7, T8
{c, j}	3	T4, T6, T7
{b, j}	2	T6, T7

{m,b}, {c,b}, {c,j}

Count the support for 3-itemsets

Itemset	Count	Transactions
{m, c, b}	2	T1, T6
{m, c, j}	1	T6
{m, b, j}	1	T6
{c, b, j}	2	T6, T7

Nothing, all below 3

Applications - Offline Stores

Offline Stores

- Soaps on floor 1, Towels on floor 10
- High occurrence of baskets with both soaps and towels.

Strategies for store management

- On-the spot sales – Place some towels and bathing accessories with soaps on floor 1
- Pricing strategy – Put discount on soaps and raise price of towels

Applications - Online Commerce

- **Online – E-Retail like E-bay or Amazon**
- Scale – Millions of items and customers
- Can tailor offer even for individual customers
- **Market based strategy –**
  - Each basket represents items bought by a specific customer
  - Recommended items that other customers with similar basket purchased
- **Collaborative filtering –**
  - Finding customers with similar purchase
  - Recommend items based on what other customers bought but this customer hasn’t.
- Scale implication
  - Even few instances of same itemset can be of value
  - Because offers can be personalised even for a single customer
  - Need Millions of pairs for diverse recommendation

Applications – Brick and Mortar Stores

- **Traditional street-side businesses**
- Scale – Limited by physical space and location
- **Market based strategy –**
  - Identify popular combinations
  - Organise store layout or promotions based on these combinations
- **Bulk Strategy**
  - Focus on items combinations bought by vast numbers
- Scale implication
  - Need thousands of occurrences of same interest to take action
  - Limited number of promotions or rearrangement possible due to physical constraints

Other Applications

- Customer transaction analysis
- Other data mining problems
  - Classification, clustering, outlier analysis
- Web mining
  - Processes web logs to determine browsing behaviour patterns.
  - Applications: Website design optimization, Making user-specific recommendations
- Software bug analysis
- Chemical and Biological analysis
  - Drug design
  - Genetic research

Association Rule Mining

Association Rule Mining

- Objective: Discover rules that indicate how certain items in a dataset relate to other items
- An Association Rule is typically represented as $I \rightarrow j$
  - Implying that if items in I appear in a transaction, j is likely to appear too

Formal Definition:

- $I = \{I_1, I_2, ..., I_m\}$: Set of m distinct attributes or items.
- D : Database of transactions. Each transaction T is a set of items from I.
- Rule: $X \Rightarrow Y$, where both X and Y are itemsets from I, and $X \cap Y = \emptyset$.
  - X is the antecedent, and Y is the consequent.

Criteria for Rule Selection:

- Support – measures the rule's overall popularity in the dataset
- Confidence – Measures the rule's reliability

Association Rule Mining

Criteria for Rule Selection:

- Support – measures the rule's overall popularity in the dataset
- Confidence – Measures the rule's reliability

Support: A rule $X \Rightarrow Y$ is interesting if the union $X \cup Y$ is a frequent itemset.

- $support(X \cup Y) \geq \text{minimum support } s.$

Confidence: Measures the rule's reliability.

- $Confidence(X \Rightarrow Y) = \frac{support(X \cup Y)}{support(X)}.$
- Represents the likelihood that Y will appear in a transaction given X is present.
- A rule is deemed interesting if its confidence exceeds a minimum confidence threshold c

Example

Baskets

1. $B_1 = \{m, c, b\}$
2. $B_2 = \{m, p, j\}$
3. $B_3 = \{m, b\}$
4. $B_4 = \{c, j\}$
5. $B_5 = \{m, b, p\}$
6. $B_6 = \{m, c, b, j\}$
7. $B_7 = \{c, b, j\}$
8. $B_8 = \{m, b, c\}$

An association rule $\{m, b\} \rightarrow$ c has Support = Frequency$\{m,b,c\} = \frac{3}{8} = 37.5\%$

$$\text{Confidence} = \frac{\text{Support}\{m,b,c\}}{\text{Support}\{m,b\}} = \frac{3}{5} = 60\%$$

Association Rule Mining

- Confidence can be easily derived from A and A ∪ B.
- Once the support counts of A, B, and A ∪ B it is straightforward to derive the association rules A → B and B → A.
- Thus, problem of mining association rules can be reduced to – mining frequent itemsets

This can be viewed as two-step process

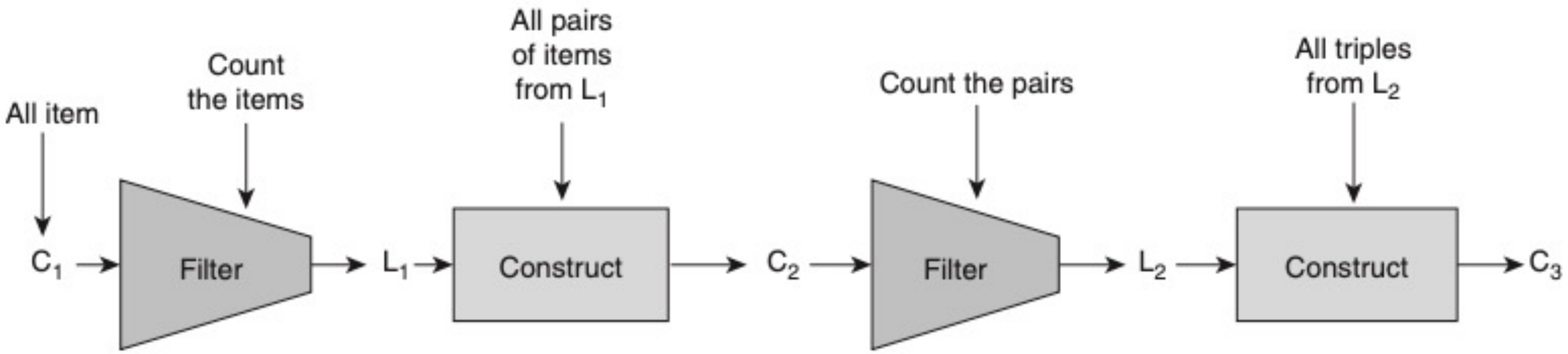
- **Find all frequent itemsets**
  - Each of the itemsets will occur at least as frequently as predetermined minimum support count
  - Various algorithms – Apriori, FP-growth, Eclat etc.
- **Generate strong association rules from the frequent itemsets**
  - The rules must satisfy minimum support and minimum confidence
  - The rules are generated by creating different combinations of antecedent and consequent

Apriori Algorithm

Algorithm for finding Frequent Itemsets

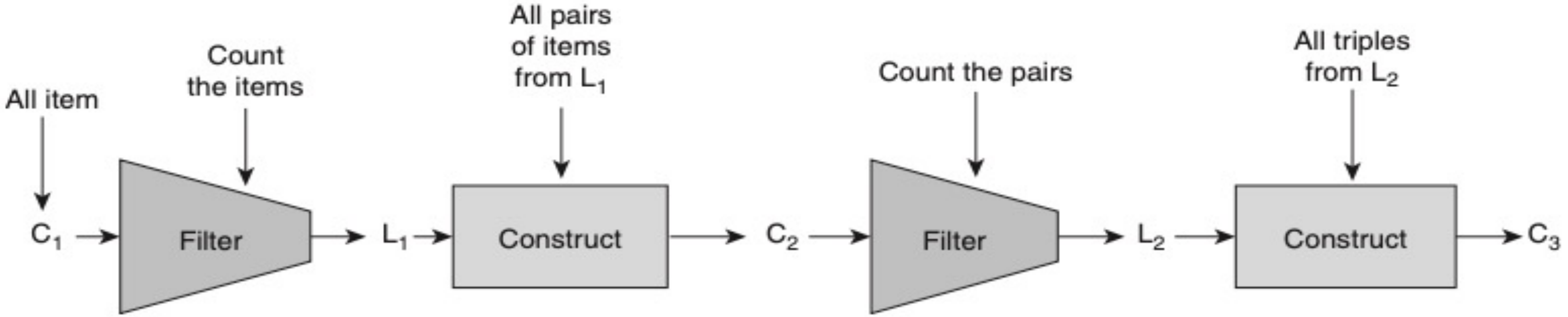
Apriori Algorithm

- Goal is to find pairs of items that occur frequently together
  - Ensuring efficient use of memory and computational resources
- Reduces the number of candidates
  - If an item is infrequent, then any of its superset will also be infrequent



Apriori Algorithm

1. **Initialization**: Define a support threshold s.
2. **Candidate Generation**: Generate candidate itemsets of size k. Initially k = 1.
3. **Counting**: For each candidate itemset, count its occurrences in the dataset.
4. **Pruning**: Eliminate the candidates that do not meet the support threshold s.
5. **Incrementing and Repeating**: Generate new candidate itemsets of size k+1 using the frequent itemsets from the previous step. Repeat the counting and pruning steps.
6. **Termination**: Stop when no more frequent itemsets can be generated.



Example - Apriori

Items – {a,b,c,d,e}
Baskets
1.{a, b}
2.{a, b, c}
3.{a, b, d}
4.{ b, c, d}
5.{a, b, c, d}
6.{a, b, d, e}

Support threshold $s = 3$.

1.

(a) Construct $C_1 = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}$.
(b) Count the support of itemsets in C_1.
(c) Remove infrequent itemsets to get $L_1 = \{ \{a\}, \{b\}, \{c\}, \{d\} \}$.

2.

(a) Construct $C_2 = \{ \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\} \}$.
(b) Count the support of itemsets in C_2.
(c) Remove infrequent itemsets to get $L_2 = \{ \{a, b\}, \{a, d\}, \{b, c\}, \{b, d\} \}$.

3.

(a) Construct $C_3 = \{ \{a, b, c\}, \{a, b, d\}, \{b, c, d\} \}$. Note that we can be more careful here with the rule generation. For example, we know $\{b, c, d\}$ cannot be frequent since $\{c, d\}$ is not frequent. That is, $\{b, c, d\}$ should not be in C_3 since $\{c, d\}$ is not in L_2.
(b) Count the support of itemsets in C_3.
(c) Remove infrequent itemsets to get $L_3 = \{ \{a, b, d\} \}$.

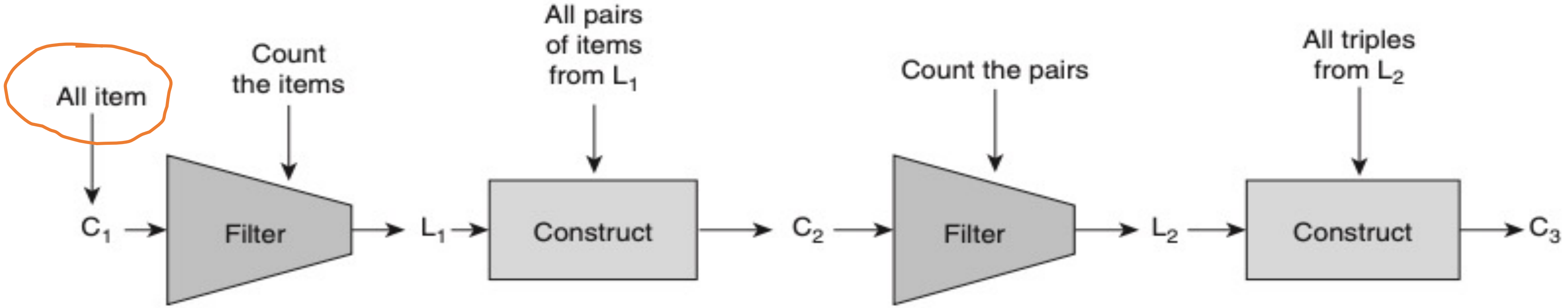
4. Construct $C_4 = \{\text{empty set}\}$.

Algorithm of Park-Chen-Yu (PCY)

Algorithm for finding Frequent Itemsets

Limitations of Apriori

- **Multiple Database Scans**: Requires multiple passes over the entire dataset.
- **Memory Consumption**: Generates large numbers of supersets.
- **Candidate Overhead**: Excessive computation for potentially non-frequent itemsets.
- **Sub-optimal Memory Use**: Does not leverage available memory efficiently.



PCY Algorithm

- Algorithm called as DHP – Direct Hashing and Pruning
- Example -

Given: Database *D*; minimum support = 2 and the following data.

TID	*Items*
1	1,3,4
2	2,3,5
3	1,2,3,5
4	2,5

Example

Given: Database D; minimum support = 2 and the following data.

TID	*Items*
1	1,3,4
2	2,3,5
3	1,2,3,5
4	2,5

Example

Pass 1:

Step 1: Scan D along with counts. Also form possible pairs and hash them to the buckets. For example, {1,3}:2 means pair {1,3} hashes to bucket 2.

(x+y) mod 3

TID	*Items*
1	1,3,4
2	2,3,5
3	1,2,3,5
4	2,5

Itemset	*Sup*
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

T1	{1,3}:2, {1,4}:1, {3,4}:3
T2	{2,3}:1 , {2,5}:3, {3,5}:5
T3	{1,2}:4, {1,3}:2, {1,5}:5, {2,3}:1, {2,5}:3, {3,5}:5
T4	{2,5}:3

Step 2: Using the hash function as discussed in step 1 the bucket looks like the one shown below.

Bucket	**1**	**2**	**3**	**4**	**5**
Count	**3**	**2**	**4**	**1**	**3**

PCY Algorithm

- **Hash-based Bucket Counting**: Reduces candidate pairs using hash mechanism.
- **Optimal Memory Use**: Efficiently uses memory for both item counts and hash table of pairs.
- **Candidate Reduction**: Minimized computational effort by pruning many item pairs.
- **Efficient Scanning**: Potentially fewer passes needed for subsequent itemsets.

Why consider PCY over Apriori

- **Memory Efficiency**: PCY utilizes available memory more effectively.
- **Reduced I/O**: Fewer database scans save processing time.
- **Pruned Candidates**: Fewer candidate itemsets reduce computational efforts.
- **Note**: Efficacy depends on data distribution and hash function quality.