DATA WAREHHOUSING LAB ASSIGNMENT

Garapati Lakshmi Poojitha
AP21110011038
CSE-P

1)Implement Apriori algorithm

```
pip install mlxtend
```

In [30]:

```python
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd
# Define the transaction data
data = [
 ['I1', 'I2', 'I5'],
 ['I2', 'I4'],
 ['I2', 'I3'],
 ['I1', 'I2', 'I4'],
 ['I1', 'I3'],
 ['I2', 'I3'],
 ['I1', 'I3'],
 ['I1', 'I2', 'I3', 'I5'],
 ['I1', 'I2', 'I3']
]
# Initialize the TransactionEncoder
encoder = TransactionEncoder()
encoder_ary = encoder.fit(data).transform(data)
# Create a DataFrame with the encoded transaction data
df = pd.DataFrame(encoder_ary,
columns=encoder.columns_)
# Define the minimum support count
min_support = 2
# Initialize candidate itemsets
C = df.columns.tolist()
L = []
# Step 1: Generating 1-itemset Frequent Pattern
print("Itemset Sup.Count")
for item in C:
 sup_count = df[item].sum()
 if sup_count >= min_support:L.append([item])
 print(f"{{{item}}} \t {sup_count}")
# Step 2: Generating 2-itemset Frequent Pattern
print("\nStep 2: Generating 2-itemset Frequent Pattern")
C2 = []
print("Itemset \t Sup.Count")
for i in range(len(L)):
 for j in range(i + 1, len(L)):
    itemset = [L[i][0],L[j][0]]
    sup_count =(df[L[i][0]] & df[L[j][0]]).sum()
    if sup_count >= min_support:
        C2.append(itemset)
        print(f"{set(itemset)} \t {sup_count}")
# Step 3: Generating 3-itemset Frequent Pattern
print("\nStep 3: Generating 3-itemset Frequent Pattern")
C3 = []
print("Itemset \t Sup.Count")
for i in range(len(C2)):
    for j in range(i + 1, len(C2)):
        itemset = list(set(C2[i] + C2[j]))
        if len(itemset) == 3:
            sup_count = (df[C2[i][0]] & df[C2[i][1]] & df[C2[j][0]] & df[C2[j][1]
            if sup_count >= min_support and itemset not in C3:
                C3.append(itemset)
                print(f"{set(itemset)} \t {sup_count}")
```

```
Itemset Sup.Count
{I1}
        6
{I2}
```

7

{I3}

6

{I4}

2

{I5}

2

Step 2: Generating 2-itemset Frequent Pattern

Itemset Sup.Count

{'I1', 'I2'} 4

{'I3', 'I1'} 4

{'I5', 'I1'} 2

{'I3', 'I2'} 4

{'I4', 'I2'} 2

{'I5', 'I2'} 2

Step 3: Generating 3-itemset Frequent Pattern

Itemset Sup.Count

{'I2', 'I3', 'I1'} 2

{'I5', 'I1', 'I2'} 2

Q2.Generation of the candidate itemsets and frequent itemsets where the minimum support count is 2.

In [34]:
```python
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

# Define the transaction data
data = [
    ['I1', 'I2', 'I5'],
    ['I2', 'I4'],
    ['I2', 'I3'],
    ['I1', 'I2', 'I4'],
    ['I1', 'I3'],
    ['I2', 'I3'],
    ['I1', 'I3'],
    ['I1', 'I2', 'I3', 'I5'],
    ['I1', 'I2', 'I3']
]

# Initialize the TransactionEncoder
encoder = TransactionEncoder()
encoder_ary = encoder.fit(data).transform(data)

# Create a DataFrame with the encoded transaction data
df = pd.DataFrame(encoder_ary, columns=encoder.columns_)

# Define a lower minimum support count
min_support = 0.2  # Adjust this value as needed

# Use Apriori algorithm to generate frequent itemsets
frequent_itemsets = apriori(df, min_support=min_support, use_colnames=True)

# Display the frequent itemsets
print("Frequent Itemsets:")
print(frequent_itemsets)

# Generate association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)

# Display the association rules
print("\nAssociation Rules:")
print(rules)
```

```
Frequent Itemsets:
support
itemsets 0
0.666667
(I1)
1 0.777778 (I2)
2 0.666667 (I3)
3 0.222222 (I4)
4 0.222222 (I5)
5 0.444444 (I1, I2)
6 0.444444 (I3, I1)
7 0.222222 (I5, I1)
8 0.444444 (I3, I2)
9 0.222222 (I4, I2)
10 0.222222 (I5, I2)
11 0.222222 (I3, I1, I2)
```

```
12 0.222222 (I5, I1, I2)

Association Rules: antecedents consequents antecedent
 support consequent support support \
   0 (I3) (I1) 0.666667  0.666667 0.444444
   1 (I1) (I3) 0.666667  0.666667 0.444444
   2 (I5) (I1) 0.222222  0.666667 0.222222
   3 (I1) (I5) 0.666667  0.222222 0.222222
   4 (I4) (I2) 0.222222  0.777778 0.222222
   5 (I2) (I4) 0.777778  0.222222 0.222222
   6 (I5) (I2) 0.222222  0.777778 0.222222
   7 (I2) (I5) 0.777778  0.222222 0.222222
   8 (I5, I1) (I2) 0.222222  0.777778 0.222222
   9 (I5, I2) (I1) 0.222222  0.666667 0.222222
  10 (I1, I2) (I5) 0.444444  0.222222 0.222222
 11 (I5) (I1, I2) 0.222222  0.444444 0.222222
 12 (I1) (I5, I2) 0.666667  0.222222 0.222222
 13 (I2) (I5, I1) 0.777778  0.222222 0.222222

  confidence lift leverage conviction
zhangs_metric 0 0.666667 1.000000
0.000000 1.000000 0.000000
1 0.666667 1.000000 0.000000 1.000000
    0.000000
2 1.000000 1.500000 0.074074 inf  0.428571
3 0.333333 1.500000 0.074074 1.166667
    1.000000
4 1.000000 1.285714 0.049383 inf  0.285714
5 0.285714 1.285714 0.049383 1.088889
    1.000000
6 1.000000 1.285714 0.049383 inf  0.285714
7 0.285714 1.285714 0.049383 1.088889
    1.000000
8 1.000000 1.285714 0.049383 inf  0.285714
9 1.000000 1.500000 0.074074 inf  0.428571
10 0.500000 2.250000 0.123457 1.555556
    1.000000
11 1.000000 2.250000 0.123457 inf  0.714286
12 0.333333 1.500000 0.074074 1.166667
    1.000000
13 0.285714 1.285714 0.049383 1.088889
    1.000000
```

In [ ]: