```
import itertools

# Defining the dataset
dataset = {
    'T100': {'I1', 'I2', 'I5'},
    'T200': {'I2', 'I4'},
    'T300': {'I2', 'I3'},
    'T400': {'I1', 'I2', 'I4'},
    'T500': {'I1', 'I3'},
    'T600': {'I2', 'I3'},
    'T700': {'I1', 'I3'},
    'T800': {'I1', 'I2', 'I3', 'I5'},
    'T900': {'I1', 'I2', 'I3'}
}

# Input of Minimum Support
min_sup = int(input('Enter the minimum support count: '))
print(f'The value of min_sup_count is: {min_sup}')
```

```
     Enter the minimum support count: 2
     The value of min_sup_count is: 2
```

```
# Generating Ck and Lk
def generateCkandLk(min_sup, dataset):
  start = set({})
  for i in dataset:
    start = start.union(dataset[i])
  C = {}
  L = {}
  k = 1
  while (True):
    C[f'C{k}'] = {}
    L[f'L{k}'] = {}
    if k == 1:
      for i in start:
        C[f'C{k}'][f'{set([i])}'] = 0
        for j in dataset:
          if set([i]).issubset(dataset[j]):
            C[f'C{k}'][f'{set([i])}'] += 1
    else:
      newStart = set()
      for i in range(len(start)):
        for j in range(i+1, len(start)):
          if len([frozenset(start[i].union(start[j]))][0]) == k:
            newStart = newStart.union(set([frozenset(start[i].union(start[j]))]))
      for i in newStart:
        C[f'C{k}'][f'{set(list(i))}'] = 0
        for j in dataset:
          if set(list(i)).issubset(dataset[j]):
            C[f'C{k}'][f'{set(list(i))}'] += 1
    start = []
    for i in C[f'C{k}']:
      if C[f'C{k}'][i] >= min_sup:
        L[f'L{k}'][i] = C[f'C{k}'][i]
        start.append(set(''.join(''.join(''.join(''.join(''.join(i.split("{")).split("}"))
    k += 1
    if len(L[f'L{k-1}']) == 0:
      break
  return C, L

C, L = generateCkandLk(min_sup, dataset)

print(f'The final frequent itemsets with their support counts are:\n\tL{len(L)-1} = ')
tmp = f'L{len(L)-1}'
for i in L[tmp]:
  print(f'\t\t{i}: {L[tmp][i]}')
```

```
The final frequent itemsets with their support counts are:
        L3 = 
                {'I3', 'I1', 'I2'}: 2
                {'I1', 'I2', 'I5'}: 2
```

```
def subsetsOfSpecificSize(s, n):
  return list(map(set, itertools.combinations(s, n)))

# Defining to join all the subsets
def allSubsets(LFinal):
  subsets = [[] for i in LFinal]
  itemset = 1
  for i in LFinal:
    tmp = set(''.join(''.join(''.join(''.join(''.join(i.split("{")).split("}"))).split("'"
    for j in range(1, len(tmp)+1):
      subsets[itemset-1].extend(subsetsOfSpecificSize(tmp,j))
    itemset += 1
  return subsets


subsets = allSubsets(L[f'L{len(L)-1}'])
tmp = 1
for i in subsets:
  print(f'Non-Empty Subsets of frequent itemset no. {tmp} of L{len(L)-1}:- {i[-1]}:')
  for j in i:
    print(f'\t{j}')
  tmp += 1
  print('\n')
```

```
Non-Empty Subsets of frequent itemset no. 1 of L3:- {'I3', 'I1', 'I2'}:
        {'I3'}
        {'I1'}
        {'I2'}
        {'I3', 'I1'}
        {'I3', 'I2'}
        {'I1', 'I2'}
        {'I3', 'I1', 'I2'}


Non-Empty Subsets of frequent itemset no. 2 of L3:- {'I1', 'I2', 'I5'}:
        {'I1'}
        {'I2'}
        {'I5'}
        {'I1', 'I2'}
        {'I1', 'I5'}
        {'I2', 'I5'}
        {'I1', 'I2', 'I5'}
```

```
# Taking Input of minimum confidence
min_conf = int(input('Enter the minimum confidence(in %): '))
```

```
     Enter the minimum confidence(in %): 30
```

```
# Defining the Tentative Association Rules
def tentativeRules(subsets, L):
  rules = [[] for i in L[f'L{len(L)-1}']] # from, to, conf_score
  itemset = 1
  foundSupportCountOfWholeItem = False
  supportCountOfWholeItem = 0
  for i in subsets:
    for j in i:
      if j != i[-1]:
        complementarySet = i[-1] - j
        supportCountSet = 0
        for k,l in L[f'L{len(j)}'].items():
          tmp = set(''.join(''.join(''.join(''.join(''.join(k.split("{")).split("}"))).spl
          if tmp == j:
            supportCountSet = l
            break
        if not foundSupportCountOfWholeItem:
          for k,l in L[f'L{len(L)-1}'].items():
            tmp = set(''.join(''.join(''.join(''.join(''.join(k.split("{")).split("}"))).s
            if tmp == i[-1]:
              supportCountOfWholeItem = l
              foundSupportCountOfWholeItem = True
              break
        conf_score = round(supportCountOfWholeItem*100/supportCountSet,2)
        rules[itemset-1].append([j, complementarySet, conf_score])
    foundSupportCountOfWholeItem = False
    itemset += 1
  return rules
```

```
rules = tentativeRules(subsets, L)
print('Tentative Association Rules with their confidence values are:- ')
tmp = 0
ruleNumber = 1
for i in rules:
  print(f'\n\t{subsets[tmp][-1]} of L{len(L)-1}:')
  for j in i:
    print(f'\t\tr{ruleNumber}:- {j[0]} => {j[1]}: {j[2]} %')
    ruleNumber += 1
  tmp += 1
```

```
     Tentative Association Rules with their confidence values are:-
```

```
        {'I3', 'I1', 'I2'} of L3:
                r1:- {'I3'} => {'I1', 'I2'}: 33.33 %
                r2:- {'I1'} => {'I3', 'I2'}: 33.33 %
                r3:- {'I2'} => {'I3', 'I1'}: 28.57 %
                r4:- {'I3', 'I1'} => {'I2'}: 50.0 %
                r5:- {'I3', 'I2'} => {'I1'}: 50.0 %
                r6:- {'I1', 'I2'} => {'I3'}: 50.0 %

        {'I1', 'I2', 'I5'} of L3:
                r7:- {'I1'} => {'I2', 'I5'}: 33.33 %
                r8:- {'I2'} => {'I1', 'I5'}: 28.57 %
                r9:- {'I5'} => {'I1', 'I2'}: 100.0 %
                r10:- {'I1', 'I2'} => {'I5'}: 50.0 %
                r11:- {'I1', 'I5'} => {'I2'}: 100.0 %
                r12:- {'I2', 'I5'} => {'I1'}: 100.0 %
```

```
print(f'The value of min_conf is: {min_conf} %')
```

```
The value of min_conf is: 30 %
```

```
# Defining the Final Association Rules
def finalAssociationRules(rules, min_conf):
  finalRules = [[] for i in rules]
  itemset = 1
  for i in rules:
    for j in i:
      if j[2] >= min_conf:
        finalRules[itemset-1].append([j[0],j[1],j[2]])
    itemset += 1
  return finalRules
```

```
finalRules = finalAssociationRules(rules, min_conf)
print('The Final Association Rules with their confidence percentages are:- ')
tmp = 0
ruleNumber = 1
for i in finalRules:
  print(f'\n\t{subsets[tmp][-1]} of L{len(L)-1}:')
  for j in i:
    print(f'\t\tr{ruleNumber}:- {j[0]} => {j[1]}: {j[2]} %')
    ruleNumber += 1
  tmp += 1
```

```
The Final Association Rules with their confidence percentages are:-

        {'I3', 'I1', 'I2'} of L3:
                r1:- {'I3'} => {'I1', 'I2'}: 33.33 %
                r2:- {'I1'} => {'I3', 'I2'}: 33.33 %
                r3:- {'I3', 'I1'} => {'I2'}: 50.0 %
                r4:- {'I3', 'I2'} => {'I1'}: 50.0 %
                r5:- {'I1', 'I2'} => {'I3'}: 50.0 %

        {'I1', 'I2', 'I5'} of L3:
                r6:- {'I1'} => {'I2', 'I5'}: 33.33 %
                r7:- {'I5'} => {'I1', 'I2'}: 100.0 %
                r8:- {'I1', 'I2'} => {'I5'}: 50.0 %
```

```
r9:- {'I1', 'I5'} => {'I2'}: 100.0 %
r10:- {'I2', 'I5'} => {'I1'}: 100.0 %
```

✓ 0s completed at 22:02 ● ×