In [184]:
```
import copy
import math
import gdown
import numpy as np
import pandas as pd
from collections import defaultdict
from itertools import chain, combinations
```

In [185]:
```
def def_value():
    return 0
```

In [186]:
```
def generate_candidate_set1(data):
  count = defaultdict(def_value)
  for row in data.keys():
    for item in data[row]:
      count[tuple([item])] += 1

  for item in list(count.keys()):
    if count[item]/n < support_threshold:
      del count[item]

  dict(count)
  return count
```

In [187]:
```
def generate_ordered_itemsets(c1, data):

  freq_items = []
  for item in c1.keys():
    freq_items.append([c1[item], list(item)[0]])
  freq_items.sort(reverse=True)
  freq_items_1 = []
  prev=-1
  grp=[]
  for i in range(len(freq_items)):
    if freq_items[i][0]!=prev:
      grp.sort()
      freq_items_1.extend(grp)
      grp=[]
      grp.append(freq_items[i][1])
      prev = freq_items[i][0]
    else:
      grp.append(freq_items[i][1])
  grp.sort()
  freq_items_1.extend(grp)

  for key in data.keys():
    ordered_items = []
    for val in freq_items_1:
      if val in data[key]:
        ordered_items.append(val)

    data[key] = ordered_items

  return data, freq_items_1
```

In [188]:
```
def generate_trie(data):
  # print(data)
  trie = []
  for key in data.keys():
    for i in range(len(data[key])):
      if len(trie) == i:
        trie.append({
            data[key][i]:
            {
                'supports': {
                    key
                },
                'next':[]
            }
        })
      else:
        if data[key][i] not in trie[i].keys():
          trie[i][data[key][i]] = {
              'supports':{
                  key
              },
              'next':[]
          }
        else:
          trie[i][data[key][i]]['supports'].add(key)
      if i!=0 and data[key][i] not in trie[i-1][data[key][i-1]]['next']:
          trie[i-1][data[key][i-1]]['next'].append(data[key][i])
  return trie
```

In [189]:
```
def default_val():
  return list()

def recur(trie, i, prev_path, cond_pattern_base, item):
  if i < len(trie):
    path = copy.deepcopy(prev_path)
    path.append(item)
    for next_item in trie[i][item]['next']:
      common = {}
      for j in range(len(path)):
        if len(common)<1:
          common = copy.deepcopy(set(trie[j][path[j]]['supports']))
        else:
          common = common.intersection(set(trie[j][path[j]]['supports']))
      common = common.intersection(set(trie[i+1][next_item]['supports']))
      new_support = len(list(common))
      if new_support == 0:
        continue
      cond_pattern_base[next_item].append([path,new_support])
      recur(trie, i+1, path, cond_pattern_base, next_item)

def generate_cond_pattern_base(trie):
  cond_pattern_base = defaultdict(default_val)
  if len(trie) > 0:
  # print(trie)
    for key in trie[0].keys():
      recur(trie, 0, [], cond_pattern_base, key)
  return cond_pattern_base
```

In [190]:
```
def powerset(iterable):
    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))

def get_rules_itemset(itemset, freq, confidence, data):
  rules = []
  for subset in list(powerset(itemset)):
    subset = set(subset)
    if len(subset)>0 and len(subset) != len(list(itemset)):
      count = 0
      for row in data.keys():
        flag = True
        for item in list(subset):
          if item not in data[row]:
            flag = False
            break
        if flag:
          count += 1
      # print(subset, count)
      if count == 0 or count<freq:
        continue
      cal_conf = freq / count
      if cal_conf >= confidence:
        rules.append([set(subset),set(itemset)-set(subset),cal_conf])
  return rules
```

In [191]:
```
def generator(prefix, cond_pattern_base, data, frequent_patterns):
  c1 = generate_candidate_set1(data)
  data, c1_keys = generate_ordered_itemsets(c1,data)
  trie = generate_trie(data)
  cond_pattern_base = generate_cond_pattern_base(trie)

  for item in c1_keys:

    extended_prefix = copy.deepcopy(prefix)
    extended_prefix.extend([item])
    flag = True
    for i in range(len(frequent_patterns)):
      if set(frequent_patterns[i][0]) == set(extended_prefix):
        flag = False
        break
    if flag:
      frequent_patterns.append([list(set(extended_prefix)),c1[tuple([item])]])

  if len(cond_pattern_base.keys()) < 1:
    return


  for key in cond_pattern_base.keys():
    prefix.append(key)

    data = dict()
    counter = 0
    for itemset in cond_pattern_base[key]:
      for i in range(itemset[1]):
        data['T'+str(counter)] = set(itemset[0])
        counter += 1

    generator(prefix, copy.deepcopy(cond_pattern_base), data, frequent_patterns)
```

Dataset 1 - AllElectronics.xlsx

In [192]:
```
gdown.download(url='https://drive.google.com/file/d/1W80CsLSTFUt6RhXoL4Bxcr7DNDbtDh0Z/view?usp=sharing',
output='dataset1.csv', quiet=False, fuzzy=True)
```

```
Downloading...
From: https://drive.google.com/uc?id=1W80CsLSTFUt6RhXoL4Bxcr7DNDbtDh0Z
To: /content/dataset1.csv
100%|██████████| 89.0/89.0 [00:00<00:00, 131kB/s]
```

Out[192]:
```
'dataset1.csv'
```

In [193]:
```
def default_set():
  return set()

df = pd.read_csv('dataset1.csv',header=None)

data = defaultdict(default_set)

for i in range(len(df)):
  row = 'T' + str(i)
  for j in range(len(df.iloc[i])):
    if df.iloc[i][j] is not np.nan:
      data[row].add(df.iloc[i][j])

print(data)
```

```
defaultdict(<function default_set at 0x7f01dbdfa4c0>, {'T0': {'I5', 'I1', 'I2'}, 'T1': {'I4', 'I2'}, 'T2':
{'I3', 'I2'}, 'T3': {'I4', 'I1', 'I2'}, 'T4': {'I3', 'I1'}, 'T5': {'I3', 'I2'}, 'T6': {'I3', 'I1'}, 'T7':
{'I5', 'I3', 'I1', 'I2'}, 'T8': {'I3', 'I1', 'I2'}})
```

In [194]:
```
data_copy = copy.deepcopy(data)

n = 9
support_threshold = 2/9
confidence_threshold = 0.7
support_val = 2
confidence_val = 0.7*9
```

In [195]:
```
frequent_patterns = []

c1 = generate_candidate_set1(data)

data, c1_keys = generate_ordered_itemsets(c1,data)

trie = generate_trie(data)

for key in c1_keys:
  frequent_patterns.append([[key],c1[tuple([key])]])

cond_pattern_base = generate_cond_pattern_base(trie)

for key in cond_pattern_base.keys():
  data = dict()
  prefix = [key]
  counter = 0
  for itemset in cond_pattern_base[key]:
    for i in range(itemset[1]):
      data['T'+str(counter)] = set(itemset[0])
      counter += 1

  generator(prefix, cond_pattern_base, data, frequent_patterns)
```

In [196]:
```
print('No. of freq itemsets:',len(frequent_patterns))
for i in range(len(frequent_patterns)):
  print(frequent_patterns[i][0],'=>', frequent_patterns[i][1])
```

```
No. of freq itemsets: 13
['I2'] => 7
['I1'] => 6
['I3'] => 6
['I4'] => 2
['I5'] => 2
['I1', 'I2'] => 4
['I5', 'I1'] => 2
['I5', 'I2'] => 2
['I5', 'I1', 'I2'] => 2
['I4', 'I2'] => 2
['I3', 'I1'] => 4
['I3', 'I2'] => 4
['I3', 'I1', 'I2'] => 2
```

In [197]:
```
print('Frequent itemsets of length 3:')
freq = [frequent_patterns[i] for i in range(len(frequent_patterns)) if len(frequent_patterns[i][0])==3]
for i in range(len(freq)):
  print(freq[i][0],'=>', freq[i][1])
```

```
Frequent itemsets of length 3:
['I5', 'I1', 'I2'] => 2
['I3', 'I1', 'I2'] => 2
```

In [198]:
```
# print(frequent_patterns)
req_len = 3

association_rules = []
for i in range(len(frequent_patterns)):
  if len(frequent_patterns[i][0])>1 and len(frequent_patterns[i][0])==req_len:
    association_rules.extend(get_rules_itemset(frequent_patterns[i][0], frequent_patterns[i][1], confiden
ce_threshold, data_copy))

print('\nAssociation rules which satisfy threshold confidence')
for rule in association_rules:
  print(rule[0],'=> ',rule[1], rule[2])

print('\nTotal no. of associations:',len(association_rules))
```

```
Association rules which satisfy threshold confidence
{'I5'} =>  {'I1', 'I2'} 1.0
{'I5', 'I1'} =>  {'I2'} 1.0
{'I5', 'I2'} =>  {'I1'} 1.0

Total no. of associations: 3
```

Dataset 2: Goods dataset

In [199]:
```
gdown.download(url='https://drive.google.com/file/d/15mfojAjnVzbz2ACwbRO1Q58ZqHekeZCu/view?usp=sharing',
output='dataset2.csv', quiet=False, fuzzy=True)
```

```
Downloading...
From: https://drive.google.com/uc?id=15mfojAjnVzbz2ACwbRO1Q58ZqHekeZCu
To: /content/dataset2.csv
100%|██████████| 9.96M/9.96M [00:00<00:00, 166MB/s]
```

Out[199]: 'dataset2.csv'

In [200]:
```
df = pd.read_csv('dataset2.csv')

data = defaultdict(default_set)

for i in range(100):
  row = 'T' + str(i)
  for j in range(len(df.iloc[i])):
    if not math.isnan(df.iloc[i][j]):
      data[row].add(str(int(df.iloc[i][j])))

print(data)
```

defaultdict(<function default_set at 0x7f01dbdfa4c0>, {'T0': {'32', '31', '30'}, 'T1': {'34', '35', '33'},
'T2': {'44', '46', '45', '38', '40', '39', '41', '37', '42', '36', '43'}, 'T3': {'47', '48', '39', '38'},
'T4': {'48', '57', '53', '58', '38', '49', '39', '56', '55', '54', '52', '51', '50'}, 'T5': {'60', '41',
'62', '32', '59', '61'}, 'T6': {'48', '39', '3'}, 'T7': {'63', '64', '67', '66', '65', '68'}, 'T8': {'32',
'69'}, 'T9': {'48', '72', '70', '71'}, 'T10': {'79', '77', '39', '73', '74', '76', '78', '75'}, 'T11': {'7
9', '48', '80', '38', '41', '39', '36', '81'}, 'T12': {'83', '84', '82'}, 'T13': {'85', '86', '41', '87',
'88'}, 'T14': {'48', '89', '91', '94', '39', '95', '100', '98', '93', '90', '96', '99', '101', '97', '9
2'}, 'T15': {'48', '89', '38', '39', '36'}, 'T16': {'106', '105', '41', '104', '39', '103', '107', '108',
'102'}, 'T17': {'110', '38', '41', '39', '109'}, 'T18': {'116', '112', '111', '113', '39', '117', '114',
'115', '118'}, 'T19': {'132', '120', '130', '125', '127', '131', '124', '129', '133', '126', '121', '123',
'122', '128', '119'}, 'T20': {'48', '135', '134', '136'}, 'T21': {'48', '137', '142', '147', '145', '139',
'146', '39', '141', '148', '144', '138', '140', '143', '149'}, 'T22': {'151', '39', '152', '150'}, 'T23':
{'154', '38', '39', '56', '153', '155'}, 'T24': {'48', '156', '160', '159', '157', '158'}, 'T25': {'41',
'39', '48'}, 'T26': {'162', '164', '167', '165', '161', '163', '166'}, 'T27': {'170', '48', '168', '172',
'38', '39', '173', '171', '169'}, 'T28': {'48', '176', '178', '41', '39', '32', '174', '177', '175'}, 'T2
9': {'47', '48', '179', '183', '38', '39', '181', '32', '180', '182'}, 'T30': {'184', '39', '185', '186'},
'T31': {'188', '48', '38', '41', '187', '36', '140'}, 'T32': {'48', '200', '189', '191', '186', '198', '19
3', '39', '194', '199', '196', '195', '197', '192', '190'}, 'T33': {'203', '205', '204', '208', '207', '3
9', '201', '206', '202', '209'}, 'T34': {'215', '193', '39', '213', '212', '214', '65', '211', '210'}, 'T3
5': {'217', '179', '224', '216', '220', '223', '219', '218', '221', '222'}, 'T36': {'226', '225', '227'},
'T37': {'228', '48', '231', '41', '39', '229', '230'}, 'T38': {'240', '236', '241', '38', '233', '234', '2
35', '39', '239', '232', '242', '238', '237', '36'}, 'T39': {'245', '243', '39', '244'}, 'T40': {'48', '24
7', '250', '41', '39', '248', '249', '246'}, 'T41': {'48', '39', '251', '65', '253', '252'}, 'T42': {'25
4', '48', '230'}, 'T43': {'261', '48', '258', '260', '39', '66', '78', '242', '255', '256', '257', '259'},
'T44': {'48', '39', '262'}, 'T45': {'263', '264', '267', '38', '39', '265', '36', '225', '266'}, 'T46':
{'271', '268', '269', '39', '270', '242'}, 'T47': {'79', '48', '273', '146', '39', '272', '237', '256'},
'T48': {'274'}, 'T49': {'276', '48', '281', '280', '279', '38', '39', '282', '32', '278', '275', '283', '2
77'}, 'T50': {'48', '39', '68'}, 'T51': {'48', '105', '38', '39', '95', '287', '285', '286', '96', '284'},
'T52': {'48', '292', '294', '298', '299', '289', '288', '41', '39', '297', '293', '212', '295', '296', '29
1', '290'}, 'T53': {'301', '302', '300'}, 'T54': {'303', '105', '319', '308', '312', '38', '311', '310',
'314', '36', '317', '307', '315', '304', '318', '320', '321', '305', '39', '313', '316', '306', '309'}, 'T
55': {'327', '326', '322', '324', '323', '10', '325'}, 'T56': {'48', '39', '328', '161', '152'}, 'T57':
{'329', '39', '330'}, 'T58': {'48', '337', '331', '336', '332', '333', '339', '334', '338', '335'}, 'T59':
{'48', '147', '340', '38', '18', '344', '345', '41', '343', '37', '347', '346', '341', '342'}, 'T60': {'4
8', '350', '348', '41', '39', '32', '349'}, 'T61': {'48', '353', '354', '360', '362', '364', '351', '352',
'361', '358', '359', '357', '355', '363', '356'}, 'T62': {'365', '366'}, 'T63': {'48', '60', '38', '368',
'370', '374', '41', '39', '367', '373', '371', '375', '369', '372'}, 'T64': {'48', '89', '376', '1', '11',
'385', '41', '39', '378', '384', '377', '379', '65', '381', '383', '380', '382'}, 'T65': {'387', '388', '3
89', '386'}, 'T66': {'390', '41', '38'}, 'T67': {'391', '55', '38'}, 'T68': {'397', '392', '393', '258',
'151', '340', '395', '396', '32', '398', '201', '394', '399', '43', '152'}, 'T69': {'401', '400', '404',
'403', '402', '338'}, 'T70': {'406', '39', '405', '407'}, 'T71': {'418', '258', '411', '422', '415', '48',

'416', '186', '419', '412', '408', '101', '409', '340', '421', '89', '414', '420', '413', '179', '410', '4
17'}, 'T72': {'48', '45', '39', '425', '248', '423', '424', '426'}, 'T73': {'428', '344', '141', '430', '4
29', '431', '427'}, 'T74': {'432', '39', '433', '434'}, 'T75': {'435', '48', '437', '436', '39', '438', '6
5'}, 'T76': {'337', '439', '291', '443', '48', '38', '331', '15', '440', '36', '390', '229', '449', '448',
'441', '123', '447', '446', '445', '23', '442', '444', '450'}, 'T77': {'48', '452', '451', '459', '460',
'458', '455', '454', '456', '457', '453'}, 'T78': {'48', '467', '147', '38', '462', '466', '464', '463',
'468', '470', '471', '174', '37', '461', '465', '469'}, 'T79': {'48', '473', '475', '472', '474', '39'},
'T80': {'41', '39', '476'}, 'T81': {'479', '477', '478'}, 'T82': {'485', '481', '483', '39', '484', '486',
'482', '161', '480'}, 'T83': {'48', '41', '39', '396', '32', '237', '152'}, 'T84': {'110', '105', '38', '4
87', '41', '39'}, 'T85': {'60', '381'}, 'T86': {'48', '489', '11', '496', '498', '39', '491', '500', '48
8', '499', '497', '494', '255', '493', '492', '495', '490'}, 'T87': {'39'}, 'T88': {'110', '41', '501'},
'T89': {'170', '48', '38', '502', '178', '39', '503', '32'}, 'T90': {'504', '41', '38'}, 'T91': {'507', '5
08', '509', '511', '232', '505', '506', '347', '515', '512', '225', '513', '510', '514'}, 'T92': {'170',
'48', '189', '38', '41', '39', '270', '516', '225'}, 'T93': {'48', '39'}, 'T94': {'281', '39', '517', '3
8'}, 'T95': {'519', '520', '518', '2'}, 'T96': {'522', '310', '521'}, 'T97': {'41', '523', '524'}, 'T98':
{'48', '416', '525', '527', '529', '531', '522', '530', '310', '521', '528', '526'}, 'T99': {'110', '39',
'532', '38'}})

In [201]:
```
data_copy = copy.deepcopy(data)

n = 100
support_threshold = 3/100
confidence_threshold = 0.7
support_val = 3
confidence_val = 0.7*100
```

In [202]:
```
frequent_patterns = []

c1 = generate_candidate_set1(data)

data, c1_keys = generate_ordered_itemsets(c1,data)

trie = generate_trie(data)

for key in c1_keys:
  frequent_patterns.append([[key],c1[tuple([key])]])

cond_pattern_base = generate_cond_pattern_base(trie)

for key in cond_pattern_base.keys():
  data = dict()
  prefix = [key]
  counter = 0
  for itemset in cond_pattern_base[key]:
    for i in range(itemset[1]):
      data['T'+str(counter)] = set(itemset[0])
      counter += 1

  generator(prefix, cond_pattern_base, data, frequent_patterns)
```

In [203]:
```
print('No. of freq itemsets:',len(frequent_patterns))
for i in range(len(frequent_patterns)):
  print(frequent_patterns[i][0],'=>', frequent_patterns[i][1])
```

```
No. of freq itemsets: 76
['39'] => 58
['48'] => 47
['38'] => 27
['41'] => 24
['32'] => 10
['36'] => 8
['65'] => 5
['105'] => 4
['110'] => 4
['152'] => 4
['225'] => 4
['89'] => 4
['147'] => 3
['161'] => 3
['170'] => 3
['179'] => 3
['186'] => 3
['237'] => 3
['242'] => 3
['258'] => 3
['310'] => 3
['340'] => 3
['37'] => 3
['60'] => 3
['79'] => 3
['39', '152'] => 3
['39', '38'] => 20
['48', '38'] => 15
['48', '39', '38'] => 11
['41', '39'] => 16
['48', '41'] => 13
['41', '38'] => 10
['41', '39', '38'] => 6
['48', '41', '38'] => 5
['48', '41', '39', '38'] => 3
['36', '38'] => 8
['36', '39'] => 6
['36', '48'] => 4
['36', '41'] => 3
['36', '39', '38'] => 6
```

```
['36', '41', '39', '38'] => 3
['48', '38', '41', '39', '36'] => 4
['37', '38'] => 3
['110', '38'] => 3
['110', '39'] => 3
['110', '41'] => 3
['110', '39', '38'] => 3
['105', '39'] => 4
['105', '38'] => 3
['105', '39', '38'] => 3
['39', '237'] => 3
['39', '242'] => 3
['48', '39'] => 34
['79', '39'] => 3
['170', '38'] => 3
['170', '39'] => 3
['170', '48'] => 3
['170', '39', '38'] => 3
['170', '48', '39', '38'] => 3
['170', '48', '39'] => 3
['89', '48'] => 4
['89', '39'] => 3
['89', '39', '48'] => 3
['32', '39'] => 6
['32', '48'] => 6
['32', '41'] => 4
['32', '38'] => 3
['32', '48', '39'] => 6
['32', '48', '39', '38'] => 3
['32', '48', '38'] => 3
['48', '38', '41', '39', '32'] => 3
['32', '48', '41', '38'] => 3
['48', '147'] => 3
['39', '65'] => 4
['48', '65'] => 3
['48', '39', '65'] => 3
```

In [205]:
```
print('Frequent itemsets of length 4:')
freq = [frequent_patterns[i] for i in range(len(frequent_patterns)) if len(frequent_patterns[i][0])==4]
for i in range(len(freq)):
  print(freq[i][0],'=>', freq[i][1])
```

```
Frequent itemsets of length 4:
['48', '41', '39', '38'] => 3
['36', '41', '39', '38'] => 3
['170', '48', '39', '38'] => 3
['32', '48', '39', '38'] => 3
['32', '48', '41', '38'] => 3
```

In [206]:
```
# print(frequent_patterns)
req_len = 4

association_rules = []
for i in range(len(frequent_patterns)):
  if len(frequent_patterns[i][0])>1 and len(frequent_patterns[i][0])==req_len:
    # print(frequent_patterns[i][0],frequent_patterns[i][1])
    association_rules.extend(get_rules_itemset(frequent_patterns[i][0], frequent_patterns[i][1], confiden
ce_threshold, data_copy))

print('\nAssociation rules which satisfy threshold confidence')
for rule in association_rules:
  print(rule[0],'=> ',rule[1], rule[2])

print('\nTotal no. of associations:',len(association_rules))
```

```
Association rules which satisfy threshold confidence
{'36', '41'} =>  {'39', '38'} 1.0
{'36', '41', '38'} =>  {'39'} 1.0
{'170'} =>  {'48', '39', '38'} 1.0
{'170', '48'} =>  {'39', '38'} 1.0
{'170', '39'} =>  {'48', '38'} 1.0
{'170', '38'} =>  {'48', '39'} 1.0
{'170', '48', '39'} =>  {'38'} 1.0
{'170', '48', '38'} =>  {'39'} 1.0
{'170', '39', '38'} =>  {'48'} 1.0
{'32', '38'} =>  {'48', '39'} 1.0
{'32', '48', '38'} =>  {'39'} 1.0
{'32', '39', '38'} =>  {'48'} 1.0
{'32', '41'} =>  {'48', '38'} 0.75
{'32', '38'} =>  {'48', '41'} 1.0
{'32', '48', '41'} =>  {'38'} 1.0
{'32', '48', '38'} =>  {'41'} 1.0

Total no. of associations: 16
```

In [ ]: