Chapter 14 - Exercise 2: Store data version 2

Cho dữ liệu store data trong tập tin dataset_group.csv.

Yêu cầu: Áp dụng thuật toán ECLAT để tính toán mức độ kết hợp giữa các item

1. Chuẩn hóa dữ liệu
2. Áp dụng ECLAT, Tìm kết quả
3. Cho biết 10 nhóm có độ kết hợp cao nhất?
4. Tìm kiếm thông tin từ kết quả: trong thông tin kết quả có 'eggs' không? Nếu có thì 'eggs' kết hợp với item nào?"

In [1]:

```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:

```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter14_ECLAT/'
```

In [3]:

```
import sys
from collections import defaultdict
import random
```

In [4]:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
```

In [5]:

```
# # source code from: http://codegist.net/snippet/python/eclatpy_evertheylen_pyt
def tidlists(transactions):
    tl = defaultdict(set)
    for tid, t in enumerate(transactions):
        for item in t:
            tl[item].add(tid)
    return list(tl.items())

class IntersectAll:
    def __and__(self, other):
        return other
IntersectAll = IntersectAll()
```

In [6]:
```
def eclat(items, minsup=0, minlen=1):
    frequent_itemsets = {(): IntersectAll}
    def recurse(items, prefix):
        while len(items) > 0:
            item, item_tidlist = items.pop()
            l = prefix + (item,)  # l is the (ordered) tuple of items we are loo
            new_tidlist = frequent_itemsets[prefix] & item_tidlist
            if len(new_tidlist) >= minsup:  # add frequent_itemsets to the new fi
                frequent_itemsets[l] = new_tidlist

            # define the new l-conditional database
            new_items = []
            for new_item, _item_tidlist in items:
                new_item_tidlist = _item_tidlist & item_tidlist
                if len(new_item_tidlist) >= minsup:
                    new_items.append((new_item, new_item_tidlist))

            # recurse, with l as prefix
            recurse(new_items, l)

    recurse(items.copy(), ())
    return {k: len(v) for k, v in frequent_itemsets.items() if len(k) >= minlen}
```

In [7]:
```
data = pd.read_csv("dataset_group.csv", header = None, sep=',')
```

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22343 entries, 0 to 22342
Data columns (total 3 columns):
0    22343 non-null object
1    22343 non-null int64
2    22343 non-null object
dtypes: int64(1), object(2)
memory usage: 523.8+ KB
```

In [9]:
```
data.head(3)
```

Out[9]:

	0	1	2
0	2000-01-01	1	yogurt
1	2000-01-01	1	pork
2	2000-01-01	1	sandwich bags

In [10]:
```
df = data.iloc[:,1:3]
```

In [11]:
```
df.head(3)
```

Out[11]:

	1	2
0	1	yogurt
1	1	pork
2	1	sandwich bags

In [12]:
```
dataset = df.groupby(1)[2].apply(list)
```

In [13]:
```
dataset[1]
```

Out[13]:
```
['yogurt',
 'pork',
 'sandwich bags',
 'lunch meat',
 'all- purpose',
 'flour',
 'soda',
 'butter',
 'vegetables',
 'beef',
 'aluminum foil',
 'all- purpose',
 'dinner rolls',
 'shampoo',
 'all- purpose',
 'mixes',
 'soap',
 'laundry detergent',
 'ice cream',
 'dinner rolls']
```

In [14]:
```
tl = tidlists(dataset)
tl
```

Out[14]:
```
[('yogurt',
  {0,
   1,
   4,
   6,
   11,
   19,
   22,
   24,
   25,
   28,
   30,
   31,
   32,
   33,
   34,
   35,
   38,
   44,
```

In [15]:
```
for i in range(len(tl)-1):
    if tl[i][0] == 'nan':
        print(i)
        del tl[i]
```

In [16]:
```
result = eclat(tl, minsup=150, minlen=3)
```

In [17]:
```
result
```

Out[17]:
```
{('sugar', 'eggs', 'vegetables'): 150,
 ('sugar', 'poultry', 'vegetables'): 173,
 ('sugar', 'cereals', 'vegetables'): 150,
 ('sugar', 'dishwashing liquid/detergent', 'vegetables'): 152,
 ('sugar', 'waffles', 'vegetables'): 155,
 ('sugar', 'ice cream', 'vegetables'): 151,
 ('sugar', 'dinner rolls', 'vegetables'): 157,
 ('sugar', 'vegetables', 'soda'): 155,
 ('sugar', 'vegetables', 'lunch meat'): 161,
 ('sugar', 'vegetables', 'yogurt'): 152,
 ('fruits', 'eggs', 'vegetables'): 151,
 ('fruits', 'bagels', 'vegetables'): 154,
 ('fruits', 'poultry', 'vegetables'): 150,
 ('fruits', 'dishwashing liquid/detergent', 'vegetables'): 157,
 ('fruits', 'cheeses', 'vegetables'): 151,
 ('fruits', 'ice cream', 'vegetables'): 151,
 ('fruits', 'beef', 'vegetables'): 151,
 ('fruits', 'vegetables', 'lunch meat'): 151,
 ('fruits', 'vegetables', 'yogurt'): 150,
 ('coffee/tea', 'eggs', 'vegetables'): 155
```

In [18]:
```
from collections import OrderedDict
d_sorted_by_value = OrderedDict(sorted(result.items(), key=lambda x: x[1]))
type(d_sorted_by_value)
```

Out[18]:
```
collections.OrderedDict
```

In [19]:
```
sorted_d = sorted((value, key) for (key,value) in result.items())
sorted_d[len(sorted_d)-1]
```

Out[19]:
```
(184, ('poultry', 'dinner rolls', 'vegetables'))
```

In [20]:
```
sorted_d[len(sorted_d)-10:]
```

Out[20]:
```
[(175, ('eggs', 'dishwashing liquid/detergent', 'vegetables')),
 (177, ('eggs', 'poultry', 'vegetables')),
 (178, ('eggs', 'dinner rolls', 'vegetables')),
 (178, ('poultry', 'mixes', 'vegetables')),
 (179, ('eggs', 'vegetables', 'yogurt')),
 (179, ('waffles', 'vegetables', 'lunch meat')),
 (180, ('eggs', 'vegetables', 'soda')),
 (180, ('poultry', 'vegetables', 'lunch meat')),
 (182, ('poultry', 'dishwashing liquid/detergent', 'vegetables')),
 (184, ('poultry', 'dinner rolls', 'vegetables'))]
```

In [21]:
```
# Truc quan hoa ket qua theo result vua tim ra ???
```

In [22]:

```
# "Có eggs không? nó kết hợp với item nào?"
for k, v in result.items():
    if "eggs" in k:
        print(k, ":", v)
```

```
('sugar', 'eggs', 'vegetables') : 150
('fruits', 'eggs', 'vegetables') : 151
('coffee/tea', 'eggs', 'vegetables') : 155
('paper towels', 'eggs', 'vegetables') : 163
('pasta', 'eggs', 'vegetables') : 164
('juice', 'eggs', 'vegetables') : 164
('eggs', 'bagels', 'vegetables') : 165
('eggs', 'poultry', 'vegetables') : 177
('eggs', 'ketchup', 'vegetables') : 160
('eggs', 'spaghetti sauce', 'vegetables') : 158
('eggs', 'tortillas', 'vegetables') : 151
('eggs', 'cereals', 'vegetables') : 172
('eggs', 'individual meals', 'vegetables') : 153
('eggs', 'dishwashing liquid/detergent', 'vegetables') : 175
('eggs', 'milk', 'vegetables') : 165
('eggs', 'cheeses', 'vegetables') : 171
('eggs', 'waffles', 'vegetables') : 165
('eggs', 'toilet paper', 'vegetables') : 156
('eggs', 'ice cream', 'vegetables') : 157
('eggs', 'laundry detergent', 'vegetables') : 160
('eggs', 'soap', 'vegetables') : 166
('eggs', 'mixes', 'vegetables') : 151
('eggs', 'dinner rolls', 'vegetables') : 178
('eggs', 'aluminum foil', 'vegetables') : 157
('eggs', 'beef', 'vegetables') : 160
('eggs', 'vegetables', 'butter') : 155
('eggs', 'vegetables', 'soda') : 180
('eggs', 'vegetables', 'all- purpose') : 160
('eggs', 'vegetables', 'lunch meat') : 160
('eggs', 'vegetables', 'yogurt') : 179
```

In [23]:

```
# 10 san pham ma cua hang ban nhieu nhat/it nhat (theo tl) ???
```