Chapter 13 - Exercise 2: Store data version 2

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

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

1. Chuẩn hóa dữ liệu
2. Áp dụng Apriori, Tìm kết quả
3. Tìm kiếm thông tin từ kết quả: trong thông tin kết quả có 'eggs' không? Nếu có thì 'eggs' kết hợp với item nào?"
4. Trực quan hóa dữ liệu
5. Cho biết 10 sản phẩm được mua nhiều nhất. Vẽ biểu đồ.

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter13_Apriori/
```

In [3]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
```

In [4]:
```
data = pd.read_csv("dataset_group.csv", header = None, sep=',')
```

In [5]:
```
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 [6]:
```
data.head()
```

Out[6]:

	0	1	2
0	2000-01-01	1	yogurt
1	2000-01-01	1	pork
2	2000-01-01	1	sandwich bags
3	2000-01-01	1	lunch meat
4	2000-01-01	1	all- purpose

In [7]:
```
df = data.iloc[:,1:3]
```

In [8]:
```
df.head()
```

Out[8]:

	1	2
0	1	yogurt
1	1	pork
2	1	sandwich bags
3	1	lunch meat
4	1	all- purpose

In [9]:
```
dataset = df.groupby(1)[2].apply(list)
```

In [10]:
```
dataset[1]
```

Out[10]:
```
['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 [11]:
```
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
df.head()
```

Out[11]:

	all-purpose	aluminum foil	bagels	beef	butter	cereals	cheeses	coffee/tea	dinner rolls	dishwashing liquid/detergent
0	True	True	False	True	True	False	False	False	True	False
1	False	True	False	False	False	True	True	False	False	True
2	False	False	True	False	False	True	True	False	True	False
3	True	False	False	False	False	True	False	False	False	False
4	True	False	False	False	False	False	False	False	True	False

5 rows × 38 columns

In [29]:
```
# df.info()
```

In [13]:
```
# df.isnull().any()
```

In [14]:
```
frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)
frequent_itemsets.head(3)
```

Out[14]:

	support	itemsets
0	0.374890	(all- purpose)
1	0.384548	(aluminum foil)
2	0.385426	(bagels)

In [15]:
```
frequent_itemsets.tail(3)
```

Out[15]:

	support	itemsets
49	0.305531	(vegetables, soda)
50	0.315189	(waffles, vegetables)
51	0.319579	(vegetables, yogurt)

In [16]:
```
from mlxtend.frequent_patterns import association_rules
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3)
```

Out[16]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lever
0	(aluminum foil)	(vegetables)	0.384548	0.739245	0.310799	0.808219	1.093304	0.026
1	(vegetables)	(aluminum foil)	0.739245	0.384548	0.310799	0.420428	1.093304	0.026
2	(bagels)	(vegetables)	0.385426	0.739245	0.300263	0.779043	1.053836	0.015
3	(vegetables)	(bagels)	0.739245	0.385426	0.300263	0.406176	1.053836	0.015
4	(cereals)	(vegetables)	0.395961	0.739245	0.310799	0.784922	1.061789	0.018
5	(vegetables)	(cereals)	0.739245	0.395961	0.310799	0.420428	1.061789	0.018
6	(vegetables)	(cheeses)	0.739245	0.390694	0.309043	0.418052	1.070026	0.020
7	(cheeses)	(vegetables)	0.390694	0.739245	0.309043	0.791011	1.070026	0.020
8	(dinner rolls)	(vegetables)	0.388938	0.739245	0.308165	0.792325	1.071803	0.020
9	(vegetables)	(dinner rolls)	0.739245	0.388938	0.308165	0.416865	1.071803	0.020
10	(dishwashing liquid/detergent)	(vegetables)	0.388060	0.739245	0.306409	0.789593	1.068107	0.019
11	(vegetables)	(dishwashing liquid/detergent)	0.739245	0.388060	0.306409	0.414489	1.068107	0.019
12	(eggs)	(vegetables)	0.389816	0.739245	0.326602	0.837838	1.133370	0.038
13	(vegetables)	(eggs)	0.739245	0.389816	0.326602	0.441805	1.133370	0.038
14	(vegetables)	(ice cream)	0.739245	0.398595	0.302897	0.409739	1.027957	0.008
15	(ice cream)	(vegetables)	0.398595	0.739245	0.302897	0.759912	1.027957	0.008
16	(vegetables)	(laundry detergent)	0.739245	0.378402	0.309043	0.418052	1.104783	0.029
17	(laundry detergent)	(vegetables)	0.378402	0.739245	0.309043	0.816705	1.104783	0.029
18	(lunch meat)	(vegetables)	0.395083	0.739245	0.311677	0.788889	1.067155	0.019
19	(vegetables)	(lunch meat)	0.739245	0.395083	0.311677	0.421615	1.067155	0.019
20	(poultry)	(vegetables)	0.421422	0.739245	0.331870	0.787500	1.065276	0.020
21	(vegetables)	(poultry)	0.739245	0.421422	0.331870	0.448931	1.065276	0.020
22	(vegetables)	(soda)	0.739245	0.390694	0.305531	0.413302	1.057867	0.016
23	(soda)	(vegetables)	0.390694	0.739245	0.305531	0.782022	1.057867	0.016
24	(waffles)	(vegetables)	0.394205	0.739245	0.315189	0.799555	1.081583	0.023
25	(vegetables)	(waffles)	0.739245	0.394205	0.315189	0.426366	1.081583	0.023
26	(vegetables)	(yogurt)	0.739245	0.384548	0.319579	0.432304	1.124188	0.035
27	(yogurt)	(vegetables)	0.384548	0.739245	0.319579	0.831050	1.124188	0.035

In [17]:
```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.1)
rules
```

Out[17]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	c
0	(eggs)	(vegetables)	0.389816	0.739245	0.326602	0.837838	1.133370	0.038433	
1	(vegetables)	(eggs)	0.739245	0.389816	0.326602	0.441805	1.133370	0.038433	
2	(vegetables)	(laundry detergent)	0.739245	0.378402	0.309043	0.418052	1.104783	0.029311	
3	(laundry detergent)	(vegetables)	0.378402	0.739245	0.309043	0.816705	1.104783	0.029311	
4	(vegetables)	(yogurt)	0.739245	0.384548	0.319579	0.432304	1.124188	0.035304	
5	(yogurt)	(vegetables)	0.384548	0.739245	0.319579	0.831050	1.124188	0.035304	

In [18]:
```
print(rules.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 9 columns):
antecedents           6 non-null object
consequents           6 non-null object
antecedent support    6 non-null float64
consequent support    6 non-null float64
support               6 non-null float64
confidence            6 non-null float64
lift                  6 non-null float64
leverage              6 non-null float64
conviction            6 non-null float64
dtypes: float64(7), object(2)
memory usage: 560.0+ bytes
None
```

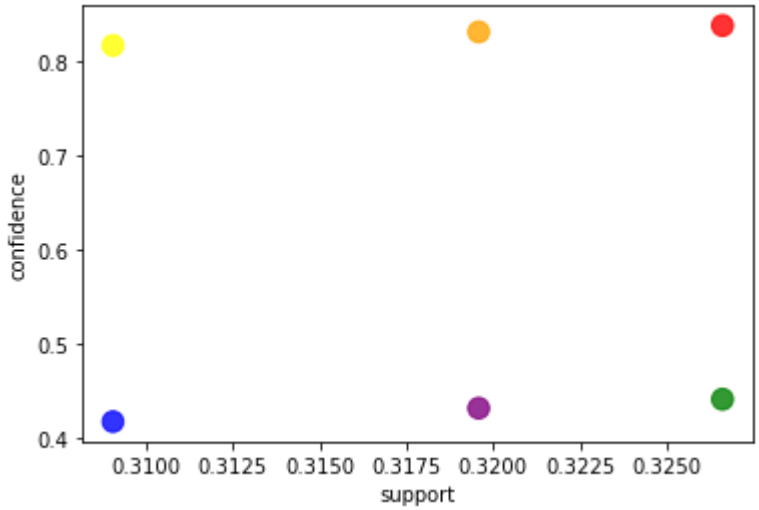
In [19]:
```
# "Có eggs không? nó kết hợp với item nào?"
for row in rules.iterrows():
    if "eggs" in row[1][0]:
        print(row)
```

```
(0, antecedents                (eggs)
consequents           (vegetables)
antecedent support        0.389816
consequent support        0.739245
support                   0.326602
confidence                0.837838
lift                       1.13337
leverage                  0.038433
conviction                 1.60799
Name: 0, dtype: object)
```

In [20]:
```
support=rules['support'].values
confidence=rules['confidence'].values
lift = rules['lift'].values
```

In [21]:
```
import matplotlib.pyplot as plt
```

In [22]:
```
plt.scatter(support, confidence, s= lift*100,alpha=0.8,
            color=['red', 'green', 'blue', 'yellow', 'purple', 'orange'])
plt.xlabel('support')
plt.ylabel('confidence')
plt.show()
```



In [23]:
```
result = df.apply(pd.value_counts).fillna(0)
result
```

Out[23]:

	all-purpose	aluminum foil	bagels	beef	butter	cereals	cheeses	coffee/tea	dinner rolls	dishwashi liquid/deterge
False	712	701	700	712	720	688	694	707	696	6
True	427	438	439	427	419	451	445	432	443	4

2 rows × 38 columns

In [24]:
```
df_true = result.iloc[1,:]
df_true[:10]
```

Out[24]:
```
all- purpose                  427
aluminum foil                 438
bagels                        439
beef                          427
butter                        419
cereals                       451
cheeses                       445
coffee/tea                    432
dinner rolls                  443
dishwashing liquid/detergent  442
Name: True, dtype: int64
```

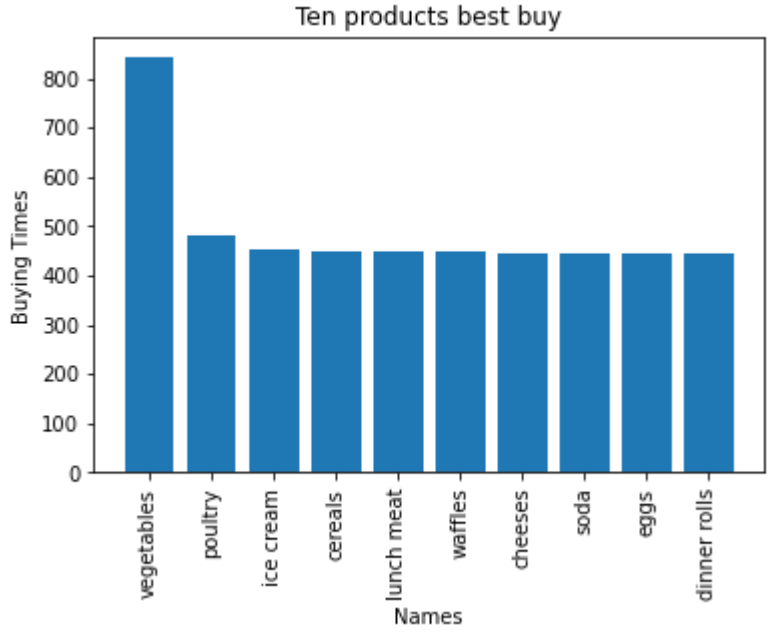
In [25]:
```
x = df_true.sort_values(ascending=False)
```

In [26]:
```
ten_products = x[:10]
ten_products
```

Out[26]:
```
vegetables      842
poultry         480
ice cream       454
cereals         451
lunch meat      450
waffles         449
cheeses         445
soda            445
eggs            444
dinner rolls    443
Name: True, dtype: int64
```

In [27]:
```
import numpy as np
pos = np.arange(len(ten_products.values))
```

In [28]:
```
plt.bar(pos, ten_products.values, align='center')
plt.xticks(pos, ten_products.keys(), rotation='vertical')
plt.ylabel('Buying Times')
plt.xlabel('Names')
plt.title('Ten products best buy')
plt.show()
```



In [ ]:
```
```