Ex 1: Store data

Cho dữ liệu store data trong tập tin store_data.

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 'milk' không? Nếu có thì 'milk' kết hợp với item nào?"
4. Trực quan hóa dữ liệu
5. Cho biết 10 sản phẩm được mua nhiều nhất, vẽ biểu đồ biểu diễn.

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

In [ ]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice_2023/Chapter11_Apriori/'
```

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

In [ ]:
```
# load dữ liệu
store_data = pd.read_csv('store_data.csv', header= None)
```

In [ ]:
```
# store_data.info()
```

In [ ]:
```
store_data.head(3)
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	1
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie	spinac
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

In [ ]:
```
records = []
for i in range(0, store_data.shape[0]):
    records.append([str(store_data.values[i,j]) for j in range(0,
                                                    store_data.shape[1])])
```

In [ ]:
```
records[0]
```

Out[8]:
```
['shrimp',
 'almonds',
 'avocado',
 'vegetables mix',
 'green grapes',
 'whole weat flour',
 'yams',
 'cottage cheese',
 'energy drink',
 'tomato juice',
 'low fat yogurt',
 'green tea',
 'honey',
 'salad',
 'mineral water',
 'salmon',
 'antioxydant juice',
 'frozen smoothie',
 'spinach',
 'olive oil']
```

In [ ]:
```
te = TransactionEncoder()
te_ary = te.fit(records).transform(records)
df = pd.DataFrame(te_ary, columns=te.columns_)
df.shape
```

Out[9]:
```
(7501, 121)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7501 entries, 0 to 7500
Columns: 121 entries,  asparagus to zucchini
dtypes: bool(121)
memory usage: 886.5 KB
```

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

Out[11]:

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	...	turkey	vegetables mix	water spray	white wine	whole weat flour	whole wheat pasta	whe whe ri
0	False	True	True	False	True	False	False	False	False	False	...	False	True	False	False	True	False	Fa
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	Fa

3 rows × 121 columns

In [ ]:
```
# df.tail()
```

In [ ]:
```
df.columns
```

Out[13]:
```
Index([' asparagus', 'almonds', 'antioxydant juice', 'asparagus', 'avocado',
       'babies food', 'bacon', 'barbecue sauce', 'black tea', 'blueberries',
       ...
       'turkey', 'vegetables mix', 'water spray', 'white wine',
       'whole weat flour', 'whole wheat pasta', 'whole wheat rice', 'yams',
       'yogurt cake', 'zucchini'],
      dtype='object', length=121)
```

In [ ]:
```
df = df.drop(['nan'], axis=1)
```

In [ ]:
```
frequent_itemsets = apriori(df, min_support=0.03, use_colnames=True)
frequent_itemsets.head()
```

Out[15]:

	support	itemsets
0	0.033329	(avocado)
1	0.033729	(brownies)
2	0.087188	(burgers)
3	0.030129	(butter)
4	0.081056	(cake)

In [ ]:
```
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	leverage	conviction
0	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
1	(frozen vegetables)	(mineral water)	0.095321	0.238368	0.035729	0.374825	1.572463	0.013007	1.218270
2	(ground beef)	(mineral water)	0.098254	0.238368	0.040928	0.416554	1.747522	0.017507	1.305401
3	(ground beef)	(spaghetti)	0.098254	0.174110	0.039195	0.398915	2.291162	0.022088	1.373997
4	(milk)	(mineral water)	0.129583	0.238368	0.047994	0.370370	1.553774	0.017105	1.209650
5	(pancakes)	(mineral water)	0.095054	0.238368	0.033729	0.354839	1.488616	0.011071	1.180529
6	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314

In [ ]:
```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.4)
rules
```

Out[17]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(chocolate)	(milk)	0.163845	0.129583	0.032129	0.196094	1.513276	0.010898	1.082736
1	(milk)	(chocolate)	0.129583	0.163845	0.032129	0.247942	1.513276	0.010898	1.111823
2	(frozen vegetables)	(mineral water)	0.095321	0.238368	0.035729	0.374825	1.572463	0.013007	1.218270
3	(mineral water)	(frozen vegetables)	0.238368	0.095321	0.035729	0.149888	1.572463	0.013007	1.064189
4	(mineral water)	(ground beef)	0.238368	0.098254	0.040928	0.171700	1.747522	0.017507	1.088672
5	(ground beef)	(mineral water)	0.098254	0.238368	0.040928	0.416554	1.747522	0.017507	1.305401
6	(ground beef)	(spaghetti)	0.098254	0.174110	0.039195	0.398915	2.291162	0.022088	1.373997
7	(spaghetti)	(ground beef)	0.174110	0.098254	0.039195	0.225115	2.291162	0.022088	1.163716
8	(mineral water)	(milk)	0.238368	0.129583	0.047994	0.201342	1.553774	0.017105	1.089850
9	(milk)	(mineral water)	0.129583	0.238368	0.047994	0.370370	1.553774	0.017105	1.209650
10	(milk)	(spaghetti)	0.129583	0.174110	0.035462	0.273663	1.571779	0.012900	1.137061
11	(spaghetti)	(milk)	0.174110	0.129583	0.035462	0.203675	1.571779	0.012900	1.093043
12	(mineral water)	(pancakes)	0.238368	0.095054	0.033729	0.141499	1.488616	0.011071	1.054100
13	(pancakes)	(mineral water)	0.095054	0.238368	0.033729	0.354839	1.488616	0.011071	1.180529
14	(mineral water)	(spaghetti)	0.238368	0.174110	0.059725	0.250559	1.439085	0.018223	1.102008
15	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314

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

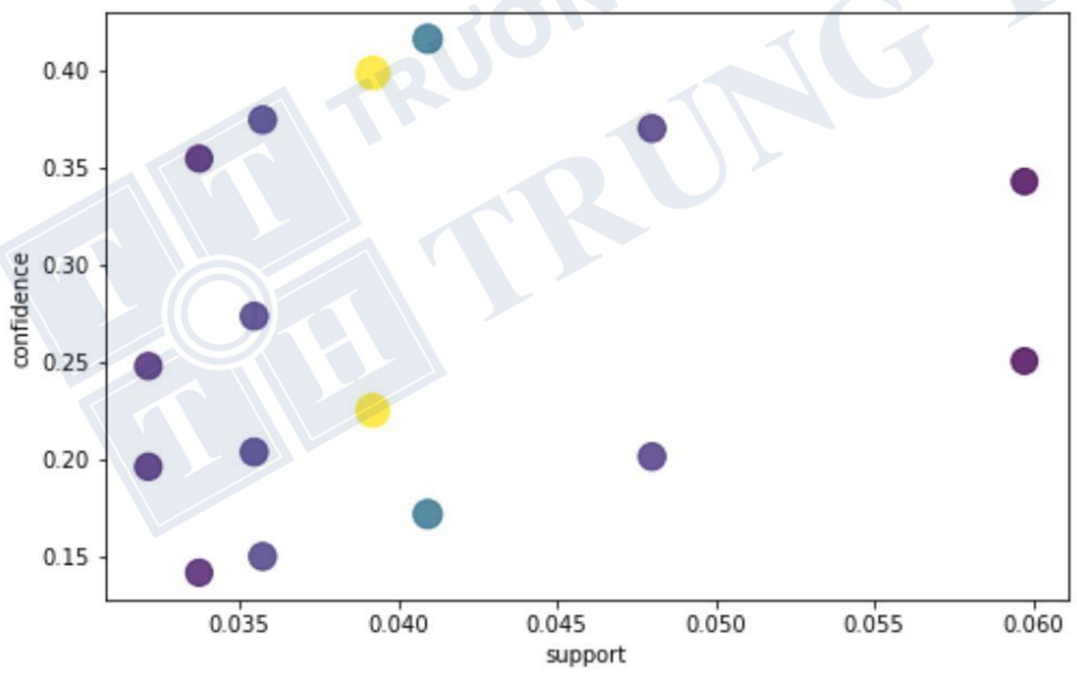
In [ ]:
```
# "Có milk không? nó kết hợp với item nào?"
for row in rules.iterrows():
    if "milk" in row[1][0]:
        print(row)
```

```
(1, antecedents                (milk)
consequents           (chocolate)
antecedent support       0.129583
consequent support       0.163845
support                  0.032129
confidence               0.247942
lift                      1.51328
leverage                0.0108976
conviction                1.11182
Name: 1, dtype: object)
(9, antecedents                    (milk)
consequents           (mineral water)
antecedent support           0.129583
consequent support           0.238368
support                     0.0479936
confidence                    0.37037
lift                          1.55377
leverage                    0.0171052
conviction                    1.20965
Name: 9, dtype: object)
(10, antecedents                 (milk)
consequents           (spaghetti)
antecedent support       0.129583
consequent support        0.17411
support                  0.0354619
confidence                0.273663
lift                      1.57178
leverage                 0.0129003
conviction                1.13706
Name: 10, dtype: object)
```

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

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

In [ ]:
```
plt.figure(figsize=(8,5))
plt.scatter(support, confidence, s= lift*100,alpha=0.8, c = lift)
plt.xlabel('support')
plt.ylabel('confidence')
plt.show()
```



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

Out[23]:

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	...	turkey	vegetables mix	water spray	white wine	whole weat flour	whole wheat pasta
False	7500	7348	7434	7466	7251	7467	7436	7420	7394	7432	...	7032	7308	7498	7377	7431	7280
True	1	153	67	35	250	34	65	81	107	69	...	469	193	3	124	70	221

2 rows × 120 columns

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

Out[24]:
```
 asparagus            1
almonds             153
antioxydant juice    67
asparagus            35
avocado             250
babies food          34
bacon                65
barbecue sauce       81
black tea           107
blueberries          69
Name: True, dtype: int64
```

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

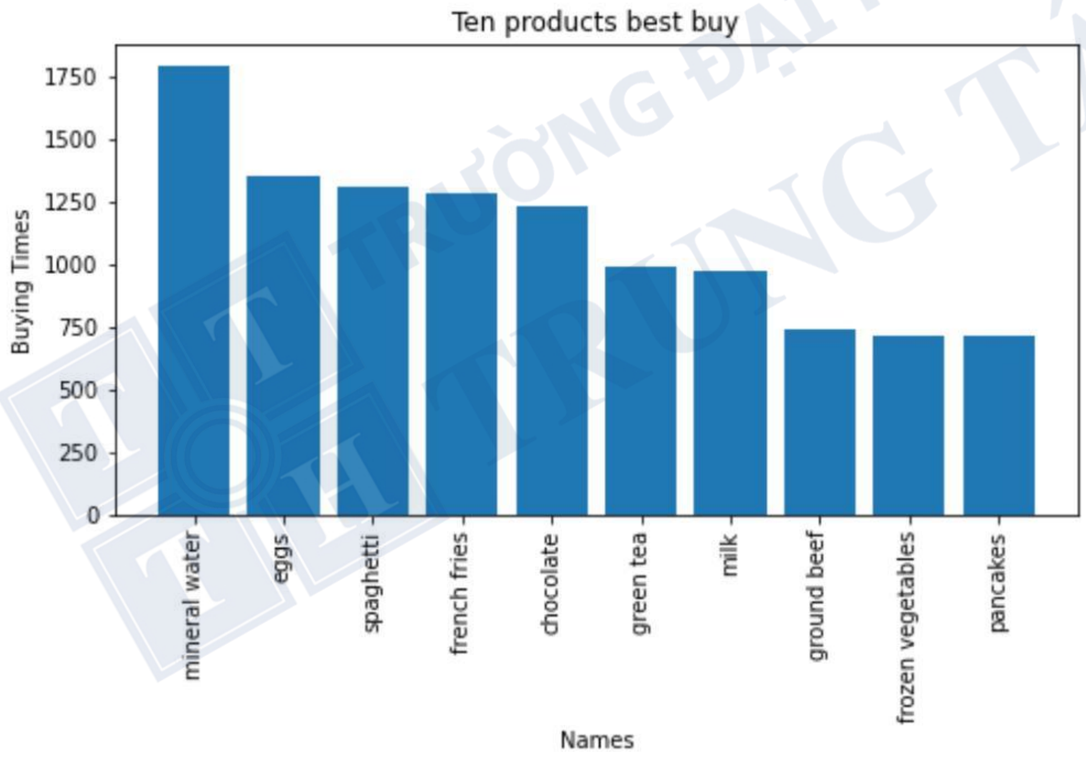
In [ ]:
```
ten_products = x[:10]
ten_products
```

Out[26]:
```
mineral water       1788
eggs                1348
spaghetti           1306
french fries        1282
chocolate           1229
green tea            991
milk                 972
ground beef          737
frozen vegetables    715
pancakes             713
Name: True, dtype: int64
```

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

In [ ]:
```
plt.figure(figsize=(8,4))
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')
```

Out[28]:
```
Text(0.5, 1.0, 'Ten products best buy')
```



In [ ]:
```
# Tinh support, confidence, lift... cua avocado -> babies food va nguoc lai
# Tinh support, confidence, lift... cua eggs -> bacon va nguoc lai
```