In [1]:
```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
!pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users\shree\anaconda3\lib\site-pac
kages (0.23.4)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\shree\anaconda3\li
b\site-packages (from mlxtend) (3.7.0)
Requirement already satisfied: scipy>=1.2.1 in c:\users\shree\anaconda3\lib\sit
e-packages (from mlxtend) (1.10.0)
Requirement already satisfied: pandas>=0.24.2 in c:\users\shree\anaconda3\lib\s
ite-packages (from mlxtend) (1.5.3)
Requirement already satisfied: joblib>=0.13.2 in c:\users\shree\anaconda3\lib\s
ite-packages (from mlxtend) (1.5.2)
Requirement already satisfied: numpy>=1.16.2 in c:\users\shree\anaconda3\lib\si
te-packages (from mlxtend) (1.23.5)
Requirement already satisfied: scikit-learn>=1.3.1 in c:\users\shree\anaconda3\
lib\site-packages (from mlxtend) (1.7.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\shree\anaconda3\lib\si
te-packages (from matplotlib>=3.0.0->mlxtend) (9.4.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\shree\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\shree\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: packaging>=20.0 in c:\users\shree\anaconda3\lib\
site-packages (from matplotlib>=3.0.0->mlxtend) (22.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\shree\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\users\shree\anaconda3\lib\sit
e-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\shree\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (1.0.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\shree\anaconda
3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\shree\anaconda3\lib\sit
e-packages (from pandas>=0.24.2->mlxtend) (2022.7)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\shree\anaconda
3\lib\site-packages (from scikit-learn>=1.3.1->mlxtend) (3.6.0)
Requirement already satisfied: six>=1.5 in c:\users\shree\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
```

In [3]:
```
import csv
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

In [6]:
```
data = []
with open('C:/Users/Shree/Market_Basket_Optimisation.csv') as file:
    reader = csv.reader(file, delimiter=',')
```

```
    for row in reader:
        data +=[row]
```

In [7]:
```
data[1:10] #list of list
```

Out[7]:
```
[['burgers', 'meatballs', 'eggs'],
 ['chutney'],
 ['turkey', 'avocado'],
 ['mineral water', 'milk', 'energy bar', 'whole wheat rice', 'green tea'],
 ['low fat yogurt'],
 ['whole wheat pasta', 'french fries'],
 ['soup', 'light cream', 'shallot'],
 ['frozen vegetables', 'spaghetti', 'green tea'],
 ['french fries']]
```

In [8]:
```
len(data)
```

Out[8]:
```
7501
```

In [9]:
```
te = TransactionEncoder()
x = te.fit_transform(data)
```

In [10]:
```
x
```

Out[10]:
```
array([[False,  True,  True, ...,  True, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False,  True, False]])
```

In [11]:
```
te.columns_
```

Out[11]:
```
[' asparagus',
 'almonds',
 'antioxydant juice',
 'asparagus',
 'avocado',
 'babies food',
 'bacon',
 'barbecue sauce',
 'black tea',
 'blueberries',
 'body spray',
 'bramble',
 'brownies',
 'bug spray',
 'burger sauce',
 'burgers',
 'butter',
 'cake',
 'candy bars',
 'carrots',
 'cauliflower',
 'cereals',
 'champagne',
 'chicken',
 'chili',
 'chocolate',
 'chocolate bread',
 'chutney',
 'cider',
 'clothes accessories',
 'cookies',
 'cooking oil',
 'corn',
 'cottage cheese',
 'cream',
 'dessert wine',
 'eggplant',
 'eggs',
 'energy bar',
 'energy drink',
 'escalope',
 'extra dark chocolate',
 'flax seed',
 'french fries',
 'french wine',
 'fresh bread',
 'fresh tuna',
 'fromage blanc',
 'frozen smoothie',
 'frozen vegetables',
 'gluten free bar',
 'grated cheese',
 'green beans',
 'green grapes',
```

```
'green tea',
'ground beef',
'gums',
'ham',
'hand protein bar',
'herb & pepper',
'honey',
'hot dogs',
'ketchup',
'light cream',
'light mayo',
'low fat yogurt',
'magazines',
'mashed potato',
'mayonnaise',
'meatballs',
'melons',
'milk',
'mineral water',
'mint',
'mint green tea',
'muffins',
'mushroom cream sauce',
'napkins',
'nonfat milk',
'oatmeal',
'oil',
'olive oil',
'pancakes',
'parmesan cheese',
'pasta',
'pepper',
'pet food',
'pickles',
'protein bar',
'red wine',
'rice',
'salad',
'salmon',
'salt',
'sandwich',
'shallot',
'shampoo',
'shrimp',
'soda',
'soup',
'spaghetti',
'sparkling water',
'spinach',
'strawberries',
'strong cheese',
'tea',
'tomato juice',
'tomato sauce',
```

```
 'tomatoes',
 'toothpaste',
 'turkey',
 'vegetables mix',
 'water spray',
 'white wine',
 'whole weat flour',
 'whole wheat pasta',
 'whole wheat rice',
 'yams',
 'yogurt cake',
 'zucchini']
```

In [12]:
```
df = pd.DataFrame(x, columns=te.columns_)
```

In [13]:
```
df
```

Out[13]:

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	ba
0	False	True	True	False	True	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	True	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
7496	False	False	False	False	False	False	False	
7497	False	False	False	False	False	False	False	
7498	False	False	False	False	False	False	False	
7499	False	False	False	False	False	False	False	
7500	False	False	False	False	False	False	False	

7501 rows × 120 columns

In [14]:
```
freq_itemset = apriori(df, min_support=0.01, use_colnames=True)
```

In [15]:
```
freq_itemset
```

Out[15]:

	support	itemsets
0	0.020397	(almonds)
1	0.033329	(avocado)
2	0.010799	(barbecue sauce)
3	0.014265	(black tea)
4	0.011465	(body spray)
...	...	...
252	0.011065	(ground beef, milk, mineral water)
253	0.017064	(spaghetti, mineral water, ground beef)
254	0.015731	(spaghetti, milk, mineral water)
255	0.010265	(spaghetti, mineral water, olive oil)
256	0.011465	(spaghetti, mineral water, pancakes)

257 rows × 2 columns

In [16]:
```
rules = association_rules(freq_itemset, metric='confidence', min_threshold=0.1
```

In [17]:
```
rules = rules[['antecedents','consequents','support','confidence']]
rules
```

Out[17]:

	antecedents	consequents	support	confidence
0	(avocado)	(mineral water)	0.011598	0.348000
1	(burgers)	(cake)	0.011465	0.131498
2	(cake)	(burgers)	0.011465	0.141447
3	(burgers)	(chocolate)	0.017064	0.195719
4	(chocolate)	(burgers)	0.017064	0.104150
...	...	...	...	...
315	(olive oil)	(spaghetti, mineral water)	0.010265	0.155870
316	(spaghetti, mineral water)	(pancakes)	0.011465	0.191964
317	(spaghetti, pancakes)	(mineral water)	0.011465	0.455026
318	(mineral water, pancakes)	(spaghetti)	0.011465	0.339921
319	(pancakes)	(spaghetti, mineral water)	0.011465	0.120617

320 rows × 4 columns

In [18]:
```
rules[rules['antecedents'] == {'cake'}]['consequents']
```

Out[18]:
```
2               (burgers)
25            (chocolate)
26                 (eggs)
29          (french fries)
30    (frozen vegetables)
32             (green tea)
35                 (milk)
37         (mineral water)
38              (pancakes)
41            (spaghetti)
Name: consequents, dtype: object
```

In [ ]: