```
import numpy as np # linear algebra
import pandas as pd # data processing
```

```
data = pd.read_csv("/content/Groceries_dataset.csv")
data.head()
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

```
data.isnull().sum()
```

```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Member_number    38765 non-null  int64 
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

```
data.shape
```

```
(38765, 3)
```

```
%pip install apyori
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5976 sha256=85e2de9
  Stored in directory: /root/.cache/pip/wheels/c4/1a/79/20f55c470a50bb3702a8cb7c94d8ada1557
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
from apyori import apriori
```

```
#encoding
products = data['itemDescription'].unique()
dummy = pd.get_dummies(data['itemDescription'])
data.drop(['itemDescription'], inplace =True, axis=1)

data = data.join(dummy)

data.head()
```

	Member_number	Date	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	ba pc
0	1808	21-07-2015	0	0	0	0	0	0	
1	2552	05-01-2015	0	0	0	0	0	0	
2	2300	19-09-2015	0	0	0	0	0	0	
3	1187	12-12-2015	0	0	0	0	0	0	
4	3037	01-02-2015	0	0	0	0	0	0	

5 rows × 169 columns

```
# Equal to SQL
# SELECT Member_number, Date, SUM(product1), SUM(product2), ..., SUM(productN)
# FROM data
# GROUP BY Member_number, Date;

data1 = data.groupby(['Member_number', 'Date'])[products[:]].sum()
```

```
# export the result to a CSV file

data1.to_csv('/content/temp-processing.csv', index=False)
```

```
# Reset the index of a Pandas DataFrame.
# The index of the DataFrame is unsorted so the reset_index() function is used to reset the inc

data1 = data1.reset_index()[products]
```

```
print("New Dimension", data1.shape)
data1.head()
```

```
New Dimension (14963, 167)
```

	tropical fruit	whole milk	pip fruit	other vegetables	rolls/buns	pot plants	citrus fruit	beef	frankfu
0	0	1	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	

5 rows × 167 columns

```
# This function is to replace numeric values with the corresponding product names.

def productnames(x):
    for product in products:
        if x[product] >0:
            x[product] = product
    return x
```

```
# The apply() method in Pandas is used to apply a function along a specific axis of a DataFrame.
# The axis=1 argument specifies that the function should be applied row-wise, i.e., to each row
# The function is to replace numeric values with the corresponding product names.

data1 = data1.apply(productnames, axis=1)
data1.head()
```

	tropical fruit	whole milk	pip fruit	other vegetables	rolls/buns	pot plants	citrus fruit	beef	frankfu
0	0	whole milk	0	0	0	0	0	0	
1	0	whole milk	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	

5 rows × 167 columns

```
# Extracting the items bought per customer
# Create and store the items into a list

x = data1.values
x = [sub[~(sub==0)].tolist()
for sub in x if sub [sub != 0].tolist()]
transactions = x
transactions[0:10]
```

```
[['whole milk', 'yogurt', 'sausage', 'semi-finished bread'],
 ['whole milk', 'pastry', 'salty snack'],
 ['canned beer', 'misc. beverages'],
 ['sausage', 'hygiene articles'],
 ['soda', 'pickled vegetables'],
 ['frankfurter', 'curd'],
```

```
 ['whole milk', 'rolls/buns', 'sausage'],
 ['whole milk', 'soda'],
 ['beef', 'white bread'],
 ['frankfurter', 'soda', 'whipped/sour cream']]
```

```
print(len(transactions))
print(transactions[0])
print(transactions[14962])
```

```
14963
['whole milk', 'yogurt', 'sausage', 'semi-finished bread']
['other vegetables', 'bottled beer']
```

```
rules = apriori(transactions, min_support = 0.00030, min_confidence = 0.05, min_lift = 3, max_
association_results = list(rules)
print(association_results[0])
```

```
RelationRecord(items=frozenset({'liver loaf', 'fruit/vegetable juice'}), support=0.0004009
```

```
print(len(association_results))
association_results[0:7]
```

```
7
[RelationRecord(items=frozenset({'liver loaf', 'fruit/vegetable juice'}),
support=0.00040098910646260775, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'liver loaf'}),
items_add=frozenset({'fruit/vegetable juice'}), confidence=0.12,
lift=3.5276227897838903)]),
 RelationRecord(items=frozenset({'ham', 'pickled vegetables'}),
support=0.0005346521419501437, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'pickled vegetables'}),
items_add=frozenset({'ham'}), confidence=0.05970149253731344, lift=3.4895055970149254)]),
 RelationRecord(items=frozenset({'roll products ', 'meat'}),
support=0.0003341575887188398, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'roll products '}), items_add=frozenset({'meat'}),
confidence=0.06097560975609757, lift=3.620547812620984)]),
 RelationRecord(items=frozenset({'misc. beverages', 'salt'}),
support=0.0003341575887188398, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'salt'}), items_add=frozenset({'misc.
beverages'}), confidence=0.05617977528089888, lift=3.5619405827461437)]),
 RelationRecord(items=frozenset({'misc. beverages', 'spread cheese'}),
support=0.0003341575887188398, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'spread cheese'}), items_add=frozenset({'misc.
beverages'}), confidence=0.05, lift=3.170127118644068)]),
 RelationRecord(items=frozenset({'soups', 'seasonal products'}),
support=0.0003341575887188398, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'soups'}), items_add=frozenset({'seasonal
products'}), confidence=0.10416666666666667, lift=14.704205974842768)]),
 RelationRecord(items=frozenset({'spread cheese', 'sugar'}),
support=0.00040098910646260775, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'spread cheese'}), items_add=frozenset({'sugar'}),
confidence=0.06, lift=3.3878490566037733)])]
```

```
for item in association_results:

    pair = item[0]
    items = [x for x in pair]

    print("Rule : ", items[0], " -> " + items[1])
    print("Support : ", str(item[1]))
    print("Confidence : ",str(item[2][0][2]))
    print("Lift : ", str(item[2][0][3]))

    print("---------------------------------")
```

```
Rule :  liver loaf  -> fruit/vegetable juice
Support :  0.00040098910646260775
Confidence :  0.12
Lift :  3.5276227897838903
---------------------------------
Rule :  ham  -> pickled vegetables
Support :  0.0005346521419501437
Confidence :  0.05970149253731344
Lift :  3.4895055970149254
---------------------------------
Rule :  roll products   -> meat
Support :  0.0003341575887188398
Confidence :  0.06097560975609757
Lift :  3.620547812620984
---------------------------------
Rule :  misc. beverages  -> salt
Support :  0.0003341575887188398
Confidence :  0.05617977528089888
Lift :  3.5619405827461437
---------------------------------
Rule :  misc. beverages  -> spread cheese
Support :  0.0003341575887188398
Confidence :  0.05
Lift :  3.170127118644068
---------------------------------
Rule :  soups  -> seasonal products
Support :  0.0003341575887188398
Confidence :  0.10416666666666667
Lift :  14.704205974842768
---------------------------------
Rule :  spread cheese  -> sugar
Support :  0.00040098910646260775
Confidence :  0.06
Lift :  3.3878490566037733
---------------------------------
```