Importing the libraries

```
!pip install apyori
```

```
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=5953 sha256=e95392981fe17670d4a2f23c8635f69ae0f768a5cf324b822bfe
  Stored in directory: /root/.cache/pip/wheels/c4/1a/79/20f55c470a50bb3702a8cb7c94d8ada15573538c7f4baebe2d
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
from mlxtend.frequent_patterns import apriori, association_rules
```

Data Processing

```
df = pd.read_csv('Market_Basket_Optimisation.csv', header = None)
basket_data = []
for i in range(0, 7501):
  basket_data.append([str(df.values[i,j]) for j in range(0, 20)])
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```

```
# Load the dataset
df = pd.read_csv('Market_Basket_Optimisation.csv', header=None)

# Initialize an empty list to store basket data
basket_data = []

# Iterate through each row in the DataFrame
for i in range(len(df)):
    # Extract each row and convert to list, skipping missing values
    basket = [str(item) for item in df.iloc[i] if not pd.isna(item)]
    basket_data.append(basket)

print(basket_data[:5])  # Print first 5 baskets to verify
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
[['shrimp', 'almonds', 'avocado', 'vegetables mix', 'green grapes', 'whole weat flour', 'yams', 'cottage cheese', 'energy drink', 'tomat
```

Training the Apriori model on the dataset

```
!pip install mlxtend
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
Requirement already satisfied: mlxtend in /usr/local/lib/python3.10/dist-packages (0.23.1)
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.25.2)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (2.0.3)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.3.2)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->mlxtend) (3.5.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxten
```

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import pandas as pd # Import pandas

# Assuming basket_data is already defined as in your previous code snippets

df = pd.DataFrame(basket_data) # Convert basket_data to DataFrame

# Create a one-hot encoded DataFrame
onehot = pd.get_dummies(df.apply(pd.Series).stack()).groupby(level=0).max() # Use groupby and max to ensure only 0 or 1

rules = apriori(onehot, min_support=0.003, use_colnames=True)
rules = association_rules(rules, metric="lift", min_threshold=3)
print(rules)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
                   antecedents                       consequents  \
0             (cottage cheese)                       (brownies)
1                   (brownies)                 (cottage cheese)
2                    (chicken)                    (light cream)
3                (light cream)                        (chicken)
4                   (escalope)            (mushroom cream sauce)
..                         ...                              ...
417  (mineral water, tomatoes)                 (spaghetti, milk)
418          (spaghetti, milk)         (mineral water, tomatoes)
419      (spaghetti, tomatoes)             (mineral water, milk)
420           (milk, tomatoes)        (mineral water, spaghetti)
421                 (tomatoes)  (mineral water, spaghetti, milk)

     antecedent support  consequent support   support  confidence      lift  \
0              0.031862            0.033729  0.003466    0.108787  3.225330
1              0.033729            0.031862  0.003466    0.102767  3.225330
2              0.059992            0.015598  0.004533    0.075556  4.843951
3              0.015598            0.059992  0.004533    0.290598  4.843951
4              0.079323            0.019064  0.005733    0.072269  3.790833
..                  ...                 ...       ...         ...       ...
417            0.024397            0.035462  0.003333    0.136612  3.852356
418            0.035462            0.024397  0.003333    0.093985  3.852356
419            0.020931            0.047994  0.003333    0.159236  3.317852
420            0.013998            0.059725  0.003333    0.238095  3.986501
421            0.068391            0.015731  0.003333    0.048733  3.097846

     leverage  conviction  zhangs_metric
0    0.002392    1.084220       0.712661
1    0.002392    1.079026       0.714038
2    0.003597    1.064858       0.844202
3    0.003597    1.325072       0.806131
4    0.004220    1.057349       0.799635
..        ...         ...            ...
417  0.002468    1.117155       0.758934
418  0.002468    1.076807       0.767641
419  0.002328    1.132311       0.713535
420  0.002497    1.234110       0.759789
421  0.002257    1.034692       0.726909

[422 rows x 10 columns]
```

Displaying the first results coming directly from the output of the Apriori function

```
results = list(rules)
results
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
['antecedents',
 'consequents',
 'antecedent support',
 'consequent support',
 'support',
 'confidence',
 'lift',
 'leverage',
      'conviction',
      'zhangs_metric']
```

Putting the results well organized into a pandas Dataframe

```
def inspect(results):
    lhs = [tuple(result['antecedents'])[0] for _, result in results.iterrows()] # Access values using column names
    rhs = [tuple(result['consequents'])[0] for _, result in results.iterrows()] # Access values using column names
    supports = [result['support'] for _, result in results.iterrows()] # Access values using column names
    confidences = [result['confidence'] for _, result in results.iterrows()] # Access values using column names
    lifts = [result['lift'] for _, result in results.iterrows()] # Access values using column names
    return list(zip(lhs, rhs, supports, confidences, lifts))

resultsinDataFrame = pd.DataFrame(inspect(rules), columns = ['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'])
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```

Displaying the results non sorted

```
resultsinDataFrame
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	cottage cheese	brownies	0.003466	0.108787	3.225330
1	brownies	cottage cheese	0.003466	0.102767	3.225330
2	chicken	light cream	0.004533	0.075556	4.843951
3	light cream	chicken	0.004533	0.290598	4.843951
4	escalope	mushroom cream sauce	0.005733	0.072269	3.790833
...	...	...	...	...	...
417	mineral water	spaghetti	0.003333	0.136612	3.852356
418	spaghetti	mineral water	0.003333	0.093985	3.852356
419	spaghetti	mineral water	0.003333	0.159236	3.317852
420	milk	mineral water	0.003333	0.238095	3.986501
421	tomatoes	mineral water	0.003333	0.048733	3.097846

422 rows × 5 columns

Displaying the results sorted by descending lifts

```
resultsinDataFrame.nlargest(n = 10, columns = 'Lift')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
342	mineral water	frozen vegetables	0.003066	0.063889	7.987176
343	frozen vegetables	mineral water	0.003066	0.383333	7.987176
330	frozen vegetables	mineral water	0.003333	0.294118	6.128268
329	mineral water	frozen vegetables	0.003333	0.069444	6.128268
175	mineral water	olive oil	0.003866	0.402778	6.115863
176	olive oil	mineral water	0.003866	0.058704	6.115863
244	frozen vegetables	mineral water	0.003200	0.139535	5.913283
241	mineral water	frozen vegetables	0.003200	0.135593	5.913283
340	mineral water	soup	0.003066	0.085821	5.646864
345	soup	mineral water	0.003066	0.201754	5.646864