```
import pandas as pd
import numpy as np
```

```
!pip install mlxtend
```

```
df=pd.read_csv("/content/Market_Basket_Optimisation.csv")
```

```
df
```

	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
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7495	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7496	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7498	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

7500 rows × 20 columns

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori,association_rules
import csv
```

```
dataset=[]
with open("/content/Market_Basket_Optimisation.csv") as file:
  reader=csv.reader(file,delimiter=',')
  for row in reader:
    dataset+=[row]
```

```
dataset
```

```
len(dataset)
```

```
7501
```

```
#transaction encoder is used to transforming data into suitable format
te=TransactionEncoder()
x=te.fit_transform(dataset)
```

```
print("The structured data is:\n",x)
```

```
The structured data is:
 [[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]]
```

```
df=pd.DataFrame(x,columns=te.columns_)
```

```
df.head()
```

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	...	turkey	vegetables mix	water spray
0	False	True	True	False	True	False	False	False	False	False	...	False	True	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False
3	False	False	False	False	True	False	False	False	False	False	...	True	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False

5 rows × 120 columns

```
#find frequent items using apriori algorithm(means support)
freq_itemset=apriori(df,min_support=0.01,use_colnames=True)
```

```
print("The frequent items(Support)in dataset are:\n",freq_itemset)
```

```
The frequent items(Support)in dataset are:
       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, mineral water, milk)
253  0.017064  (spaghetti, mineral water, ground beef)
254  0.015731         (spaghetti, mineral water, milk)
255  0.010265    (spaghetti, mineral water, olive oil)
256  0.011465     (spaghetti, pancakes, mineral water)

[257 rows x 2 columns]
```

```
rules=association_rules(freq_itemset,metric='confidence',min_threshold=0.25)
```

```
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(avocado)	(mineral water)	0.033329	0.238368	0.011598	0.348000	1.459926	0.003654	1.168147	0.325896
1	(burgers)	(eggs)	0.087188	0.179709	0.028796	0.330275	1.837830	0.013128	1.224818	0.499424
2	(burgers)	(french fries)	0.087188	0.170911	0.021997	0.252294	1.476173	0.007096	1.108844	0.353384
3	(burgers)	(mineral water)	0.087188	0.238368	0.024397	0.279817	1.173883	0.003614	1.057552	0.162275
4	(cake)	(mineral water)	0.081056	0.238368	0.027463	0.338816	1.421397	0.008142	1.151921	0.322617
...	...	...	...	...	...	...	...	...	...	...
90	(mineral water, milk)	(spaghetti)	0.047994	0.174110	0.015731	0.327778	1.882589	0.007375	1.228597	0.492451
91	(spaghetti, olive oil)	(mineral water)	0.022930	0.238368	0.010265	0.447674	1.878079	0.004799	1.378954	0.478514
92	(mineral water, olive oil)	(spaghetti)	0.027596	0.174110	0.010265	0.371981	2.136468	0.005460	1.315071	0.547034
93	(spaghetti, pancakes)	(mineral water)	0.025197	0.238368	0.011465	0.455026	1.908923	0.005459	1.397557	0.488452
94	(pancakes, mineral water)	(spaghetti)	0.033729	0.174110	0.011465	0.339921	1.952333	0.005593	1.251198	0.504819

95 rows × 10 columns

```
rules=rules[['antecedents', 'consequents',  'antecedent support','consequent support',  'support',  'confidence']]
```

```
rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence
0	(avocado)	(mineral water)	0.033329	0.238368	0.011598	0.348000
1	(burgers)	(eggs)	0.087188	0.179709	0.028796	0.330275
2	(burgers)	(french fries)	0.087188	0.170911	0.021997	0.252294
3	(burgers)	(mineral water)	0.087188	0.238368	0.024397	0.279817
4	(cake)	(mineral water)	0.081056	0.238368	0.027463	0.338816