Import neccessery libraries

In [1]:

```
!pip install apyori
```

```
Requirement already satisfied: apyori in c:\users\akarsh\anaconda3\lib\site
-packages (1.1.2)
```

In [2]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from apyori import apriori as apr
from mlxtend.frequent_patterns import apriori,association_rules
from mlxtend.preprocessing import TransactionEncoder
from scipy.special import comb
import scipy as sp
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from itertools import combinations,permutations
```

Problem

Prepare rules for the all the data sets

1) Try different values of support and confidence. Observe the change in number of rules for different support,confidence values

2) Change the minimum length in apriori algorithm

3) Visulize the obtained rules using different plots

Import data

In [11]:

```
book_data = pd.read_csv('book.csv')
book_data
```

Out[11]:

	ChildBks	YouthBks	CookBks	DoItYBks	RefBks	ArtBks	GeogBks	ItalCook	ItalAtlas	Ital
0	0	1	0	1	0	0	1	0	0	
1	1	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	1	1	1	0	1	0	1	0	0	
4	0	0	1	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	
1995	0	0	1	0	0	1	1	1	0	
1996	0	0	0	0	0	0	0	0	0	
1997	0	0	0	0	0	0	0	0	0	
1998	0	0	1	0	0	0	0	0	0	

	ChildBks	YouthBks	CookBks	DoItYBks	RefBks	ArtBks	GeogBks	ItalCook	ItalAtlas	Ital
1999	0	0	0	0	0	0	0	0	0	

Data understanding

In [12]:
```
book_data.shape
```

Out[12]:
```
(2000, 11)
```

In [13]:
```
book_data.isna().sum()
```

Out[13]:
```
ChildBks     0
YouthBks     0
CookBks      0
DoItYBks     0
RefBks       0
ArtBks       0
GeogBks      0
ItalCook     0
ItalAtlas    0
ItalArt      0
Florence     0
dtype: int64
```

In [14]:
```
book_data.dtypes
```

Out[14]:
```
ChildBks     int64
YouthBks     int64
CookBks      int64
DoItYBks     int64
RefBks       int64
ArtBks       int64
GeogBks      int64
ItalCook     int64
ItalAtlas    int64
ItalArt      int64
Florence     int64
dtype: object
```

In [15]:
```
book_data.describe().T
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
ChildBks	2000.0	0.4230	0.494159	0.0	0.0	0.0	1.0	1.0
YouthBks	2000.0	0.2475	0.431668	0.0	0.0	0.0	0.0	1.0
CookBks	2000.0	0.4310	0.495340	0.0	0.0	0.0	1.0	1.0
DoItYBks	2000.0	0.2820	0.450086	0.0	0.0	0.0	1.0	1.0
RefBks	2000.0	0.2145	0.410578	0.0	0.0	0.0	0.0	1.0
ArtBks	2000.0	0.2410	0.427797	0.0	0.0	0.0	0.0	1.0
GeogBks	2000.0	0.2760	0.447129	0.0	0.0	0.0	1.0	1.0
ItalCook	2000.0	0.1135	0.317282	0.0	0.0	0.0	0.0	1.0

	count	mean	std	min	25%	50%	75%	max
ItalAtlas	2000.0	0.0370	0.188809	0.0	0.0	0.0	0.0	1.0
ItalArt	2000.0	0.0485	0.214874	0.0	0.0	0.0	0.0	1.0

In [16]:

```
item_sets = {}
```

In [17]:

```
tran = TransactionEncoder()
tran_mode = tran.fit(book_data).transform(book_data)
```

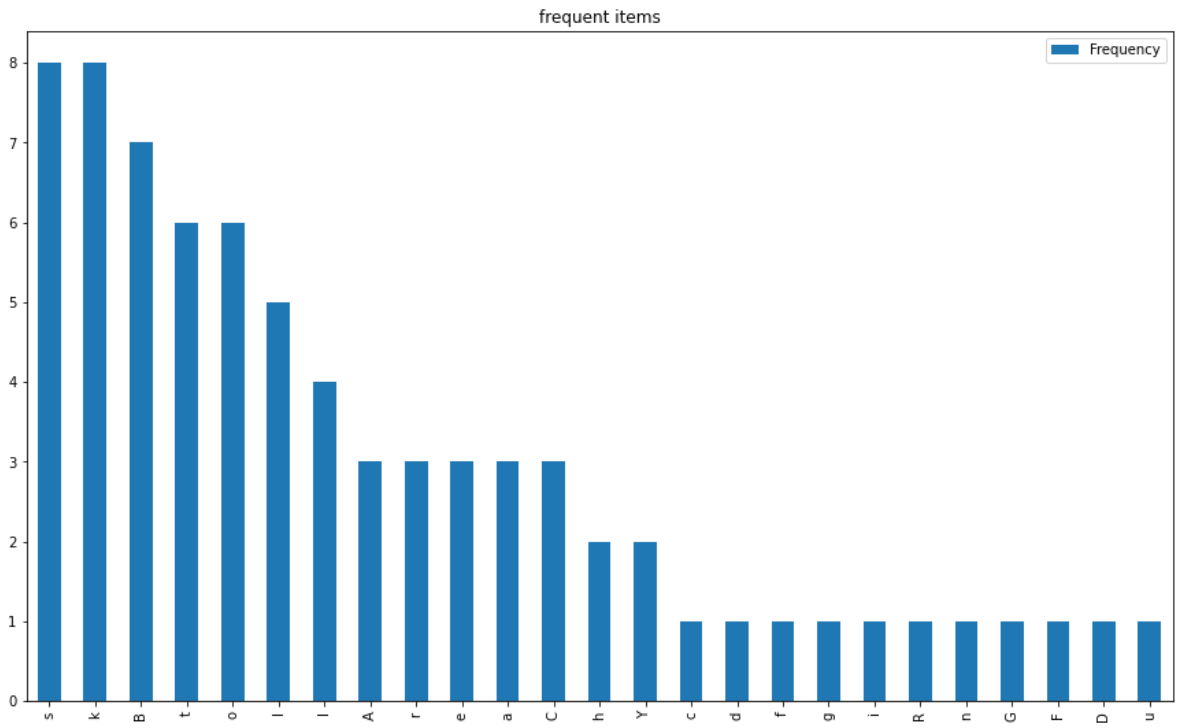
In [18]:

```
ap = pd.DataFrame(tran_mode,columns=tran.columns_)
```

In [19]:

```
ap.sum().to_frame('Frequency').sort_values('Frequency',ascending=False)[:25


plt.show()
```



Apriori algorithm

In [20]:

```
frequent_items = apriori(df = book_data,min_support=0.03,use_colnames=True)
frequent_items
```

Out[20]:

	support	itemsets
0	0.4230	(ChildBks)
1	0.2475	(YouthBks)
2	0.4310	(CookBks)
3	0.2820	(DoItYBks)
4	0.2145	(RefBks)

	support	itemsets
...	...	...
176	0.0535	(GeogBks, DoItYBks, CookBks, ChildBks, ArtBks)
177	0.0405	(GeogBks, CookBks, ChildBks, RefBks, ArtBks)
178	0.0300	(GeogBks, CookBks, ChildBks, ItalCook, ArtBks)
179	0.0370	(GeogBks, DoItYBks, CookBks, YouthBks, ArtBks)
180	0.0310	(GeogBks, DoItYBks, CookBks, ChildBks, YouthBk...

In [21]:

```
best_associates = association_rules(df = frequent_items,metric='lift',min_t
best_associates
```

Out[21]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
0	(YouthBks)	(ItalCook)	0.2475	0.1135	0.0590	0.238384	2.100298	0.03090
1	(ItalCook)	(YouthBks)	0.1135	0.2475	0.0590	0.519824	2.100298	0.03090
2	(ItalCook)	(CookBks)	0.1135	0.4310	0.1135	1.000000	2.320186	0.06458
3	(CookBks)	(ItalCook)	0.4310	0.1135	0.1135	0.263341	2.320186	0.06458
4	(DoItYBks)	(ItalArt)	0.2820	0.0485	0.0300	0.106383	2.193463	0.01632
...	...	...	...	...	...	...	...	
1399	(GeogBks)	(CookBks, DoItYBks, ChildBks, YouthBks, ArtBks)	0.2760	0.0445	0.0310	0.112319	2.524019	0.01871
1400	(DoItYBks)	(GeogBks, CookBks, ChildBks, YouthBks, ArtBks)	0.2820	0.0465	0.0310	0.109929	2.364066	0.01788
1401	(CookBks)	(GeogBks, DoItYBks, ChildBks, YouthBks, ArtBks)	0.4310	0.0335	0.0310	0.071926	2.147037	0.01656
1402	(YouthBks)	(GeogBks, CookBks, DoItYBks, ChildBks, ArtBks)	0.2475	0.0535	0.0310	0.125253	2.341169	0.01775
1403	(ArtBks)	(GeogBks, CookBks, DoItYBks, ChildBks, YouthBks)	0.2410	0.0510	0.0310	0.128631	2.522171	0.01870

1404 rows × 9 columns

In [22]:

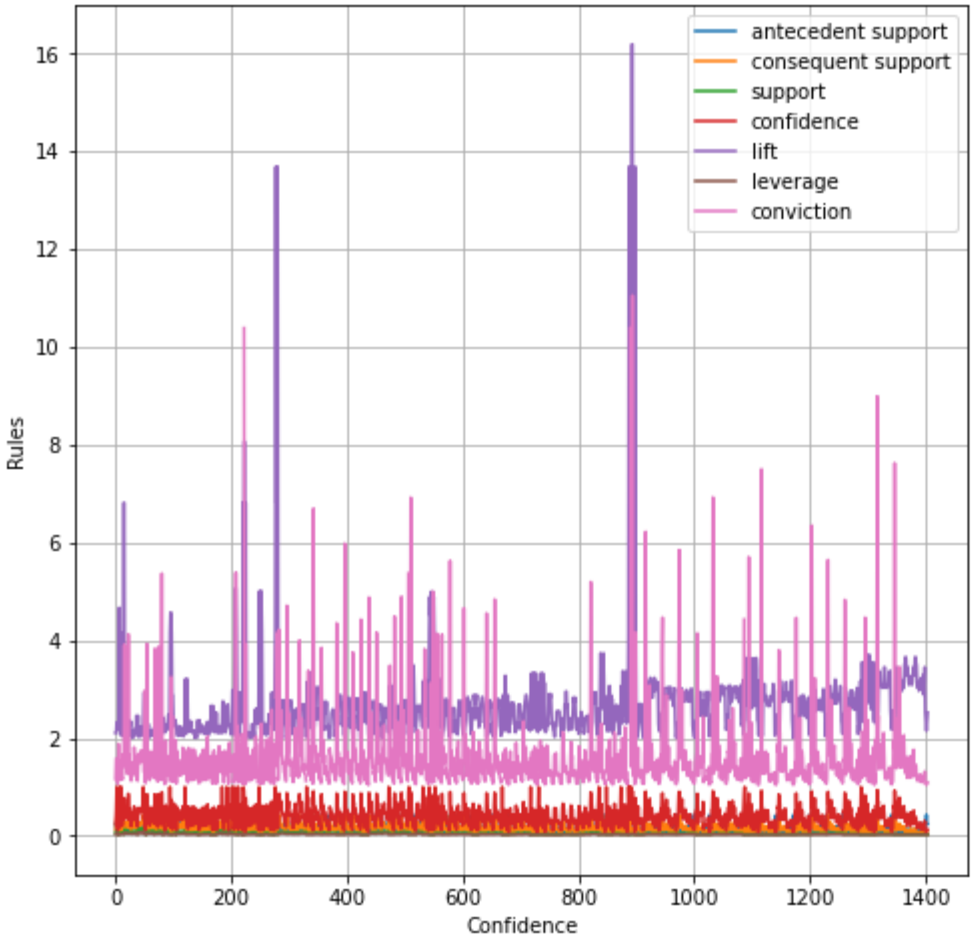
```
best_associates.shape
```

Out[22]:

```
(1404, 9)
```

In [23]:

```
best_associates.plot(figsize=(8,8),grid=True)
plt.ylabel('Rules')
plt.xlabel('Confidence')
plt.show()
```



As shown in above graph

1.Lower the Confidence level Higher the no. of rules.

2.Higher the Support, lower the no. of rules.

Lets try with Support 0.01 and Confidence at 0.5

In [24]:

```
ap_final =  apriori(ap,0.001,True)
```

In [25]:

```
rules_final = association_rules(ap_final,min_threshold=.5,support_only=Fals
```

In [26]:

```
rules_final[rules_final['confidence']> 0.6]
```

Out[26]:

antecedents	consequents	antecedent	consequent	support	confidence	lift	lever

			support	support				
0	(A)	(l)	0.0015	0.0020	0.001	0.666667	333.333333	0.000
2	(a)	(A)	0.0015	0.0015	0.001	0.666667	444.444444	0.000
3	(A)	(a)	0.0015	0.0015	0.001	0.666667	444.444444	0.000
4	(A)	(l)	0.0015	0.0025	0.001	0.666667	266.666667	0.000
5	(A)	(r)	0.0015	0.0015	0.001	0.666667	444.444444	0.000
...	...	...	...	...	...	...	...	
703	(t, Y)	(s, o, k, B)	0.0010	0.0020	0.001	1.000000	500.000000	0.000
705	(o, Y)	(s, B, k, t)	0.0010	0.0015	0.001	1.000000	666.666667	0.000
706	(B, Y)	(s, o, k, t)	0.0010	0.0010	0.001	1.000000	1000.000000	0.000
707	(k, Y)	(s, o, B, t)	0.0010	0.0010	0.001	1.000000	1000.000000	0.000
708	(Y)	(s, t, o, B, k)	0.0010	0.0010	0.001	1.000000	1000.000000	0.000

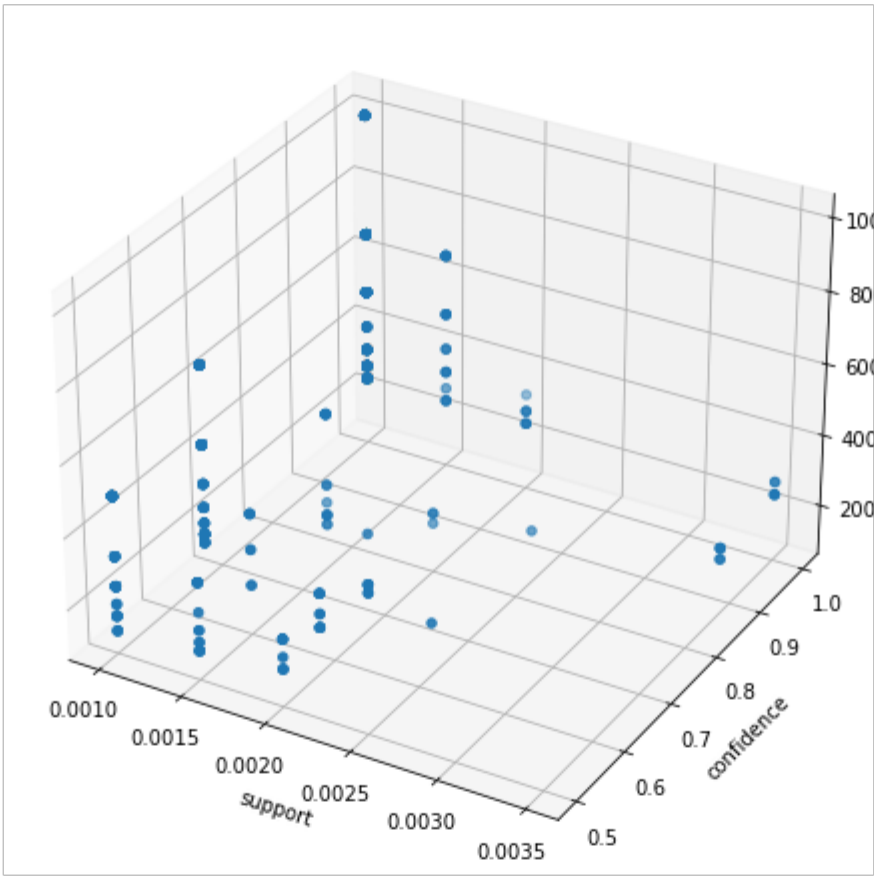
569 rows × 9 columns

In [27]:

```
support = rules_final["support"]
confidence =  rules_final["confidence"]
lift = rules_final["lift"]
```

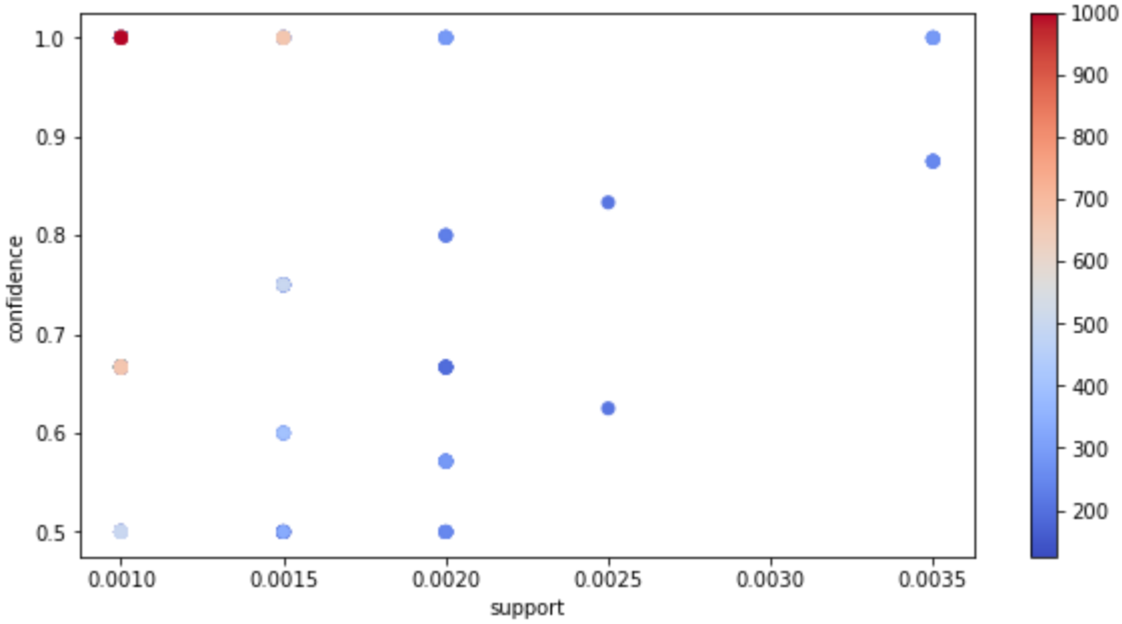
In [37]:

```
fig1 = plt.figure(figsize=(10,8))
ax1 = fig1.add_subplot(111, projection = '3d')
ax1.scatter(support,confidence,lift)
ax1.set_xlabel("support")
ax1.set_ylabel("confidence")
ax1.set_zlabel("lift")
plt.show()
```



In [34]:

```
fig1 = plt.figure(figsize=(10,5))
plt.scatter(support,confidence, c =lift, cmap = 'coolwarm')
plt.colorbar()
plt.xlabel("support");plt.ylabel("confidence")
plt.show()
```



In [ ]: