# Problem Statement: -

```
Kitabi Duniya , a famous book store in India, which was established befo
re Independence, the growth of the company was incremental year by year,
but due to online selling of books and wide spread Internet access its a
nnual growth started to collapse, seeing sharp downfalls, you as a Data
 Scientist help this heritage book store gain its popularity back and in
crease footfall of customers and provide ways the business can improve e
xponentially, apply Association Rule Algorithm, explain the rules, and v
isualize the graphs for clear understanding of solution.
```

# 1.1. Objective :-

```
Book store to gain its popularity back and increase footfall of customer
s and provide ways the business can improve exponentially, by applying A
ssociation Rule Algorithm.
```

In [6]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#book = []
#with open("D:\\360Digi\\book.csv") as f:
#    book = f.read()
book = pd.read_csv("D:\\360Digi\\book.csv")
book
```

Out[6]:

| | ChildBks | YouthBks | CookBks | DoItYBks | RefBks | ArtBks | GeogBks | ItalCook | ItalAtlas |
|---|---|---|---|---|---|---|---|---|---|
| **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 |

In [ ]:

# 3.Data Pre-processing ¶

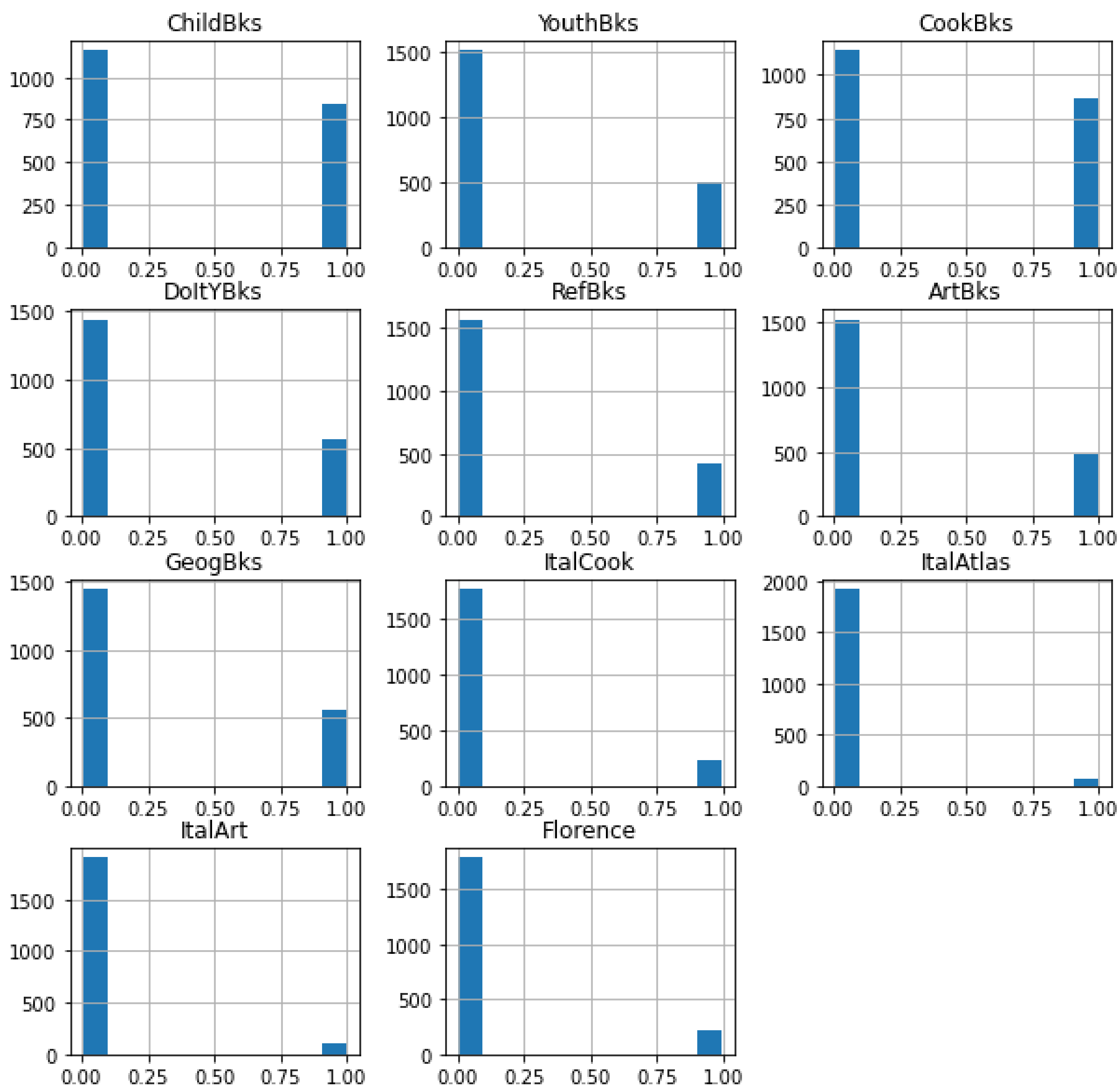
EDA

In [12]:
```
book.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ChildBks   2000 non-null   int64
 1   YouthBks   2000 non-null   int64
 2   CookBks    2000 non-null   int64
 3   DoItYBks   2000 non-null   int64
 4   RefBks     2000 non-null   int64
 5   ArtBks     2000 non-null   int64
 6   GeogBks    2000 non-null   int64
 7   ItalCook   2000 non-null   int64
 8   ItalAtlas  2000 non-null   int64
 9   ItalArt    2000 non-null   int64
 10  Florence   2000 non-null   int64
dtypes: int64(11)
memory usage: 172.0 KB
```

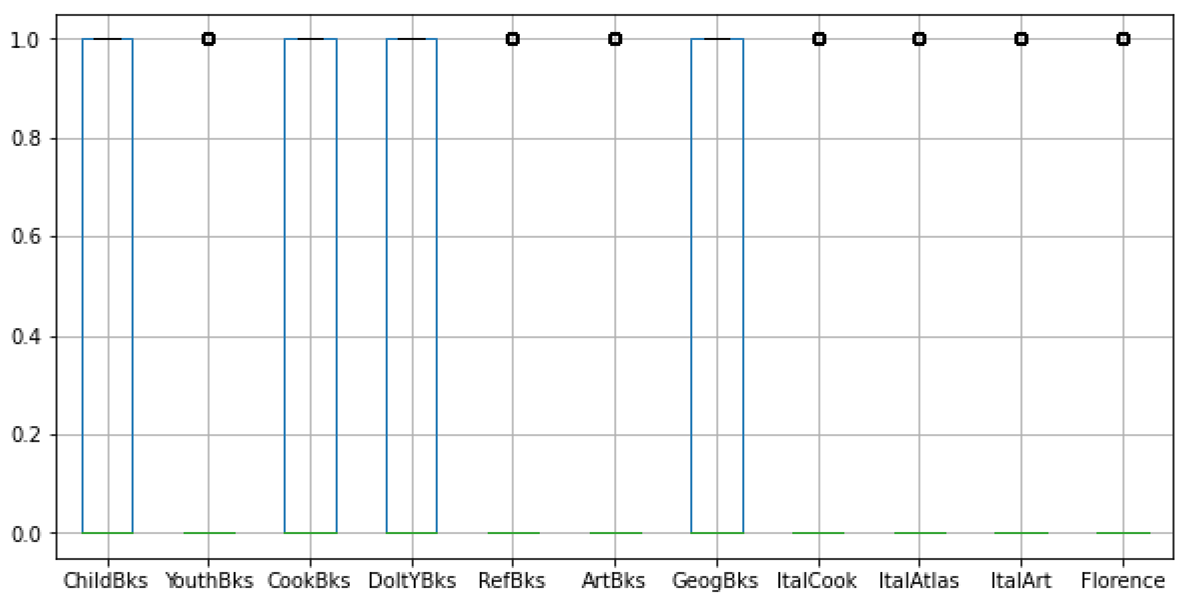
In [3]:
```
book.hist(grid=True, rwidth=0.9, figsize=(10,10))
```

Out[3]:
```
array([[<AxesSubplot:title={'center':'ChildBks'}>,
        <AxesSubplot:title={'center':'YouthBks'}>,
        <AxesSubplot:title={'center':'CookBks'}>],
       [<AxesSubplot:title={'center':'DoItYBks'}>,
        <AxesSubplot:title={'center':'RefBks'}>,
        <AxesSubplot:title={'center':'ArtBks'}>],
       [<AxesSubplot:title={'center':'GeogBks'}>,
        <AxesSubplot:title={'center':'ItalCook'}>,
        <AxesSubplot:title={'center':'ItalAtlas'}>],
       [<AxesSubplot:title={'center':'ItalArt'}>,
        <AxesSubplot:title={'center':'Florence'}>, <AxesSubplot:>]],
      dtype=object)
```

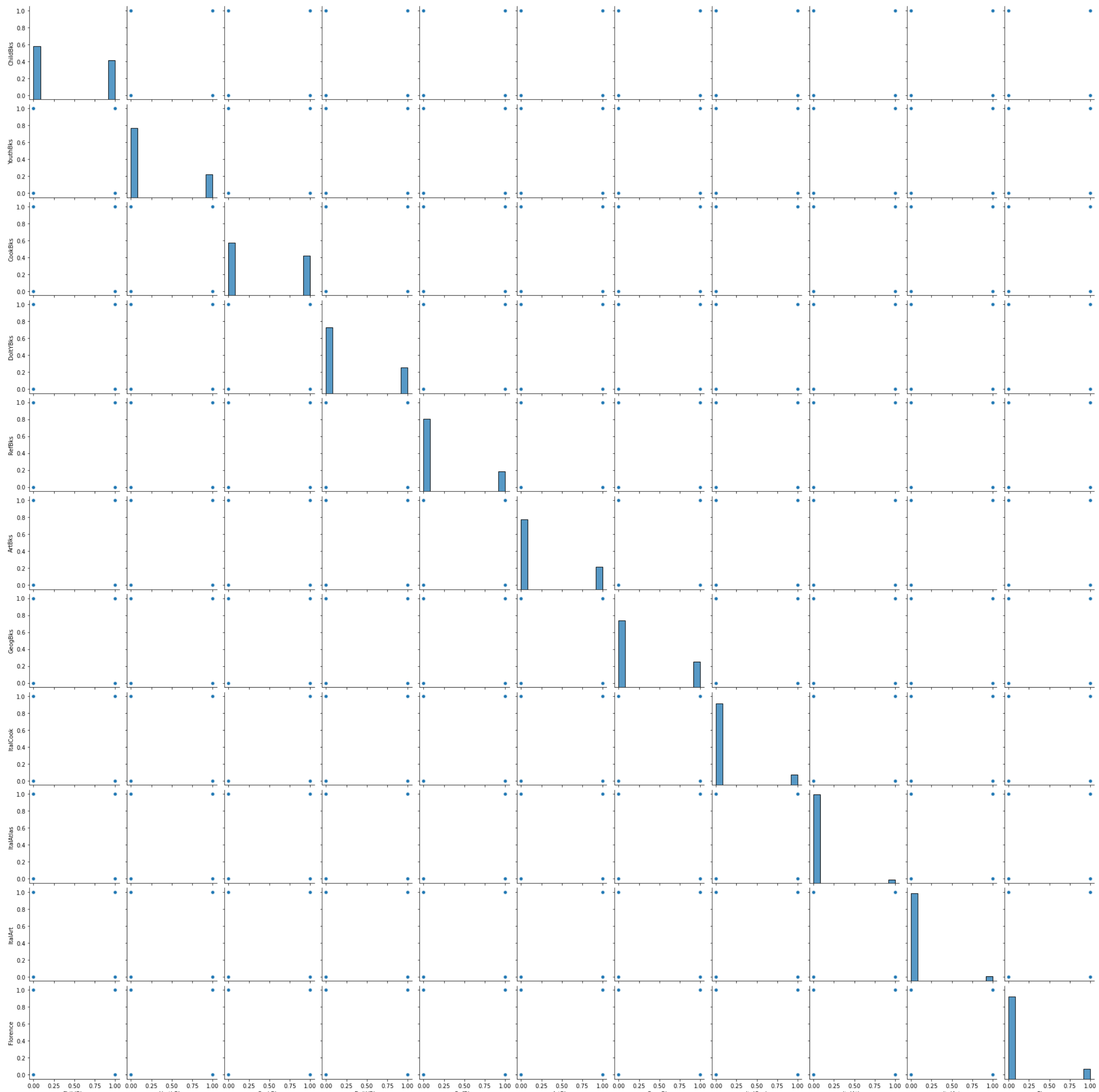


In [17]:
```
book.boxplot(grid=True,figsize=(10,5))
```

Out[17]: <AxesSubplot:>



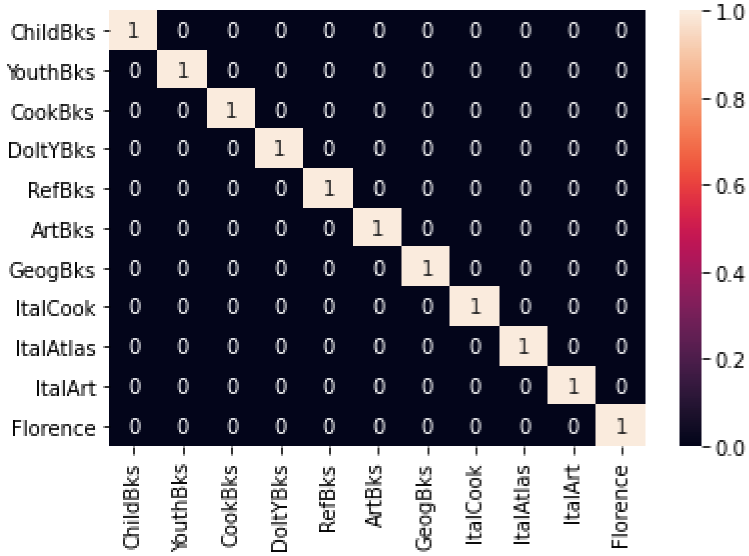
In [18]:
```
sns.pairplot(book)
plt.figure(figsize=(8,8))
plt.show()
```



```
<Figure size 576x576 with 0 Axes>
```

In [9]:
```
a = book.corr(method ='pearson')
sns.heatmap(a>0.85,annot=True)
#Since there is no correlation between variables
```

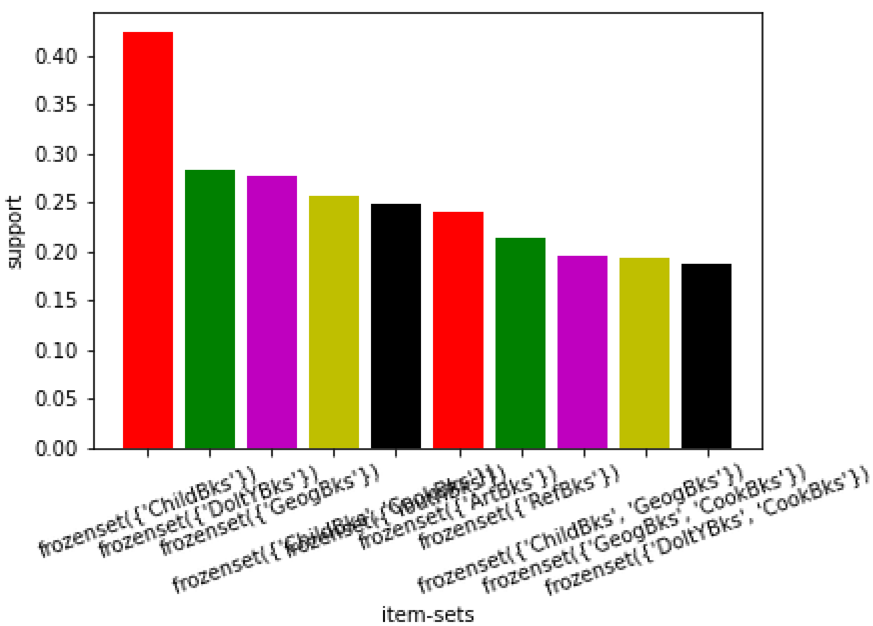
Out[9]:
```
<AxesSubplot:>
```




In [ ]:

In [4]:
```
from mlxtend.frequent_patterns import apriori, association_rules

frequent_itemsets = apriori(book, min_support = 0.05, max_len = 3, use_colnames =

# Most Frequent item sets based on support
frequent_itemsets.sort_values('support', ascending = False, inplace = True)

plt.bar(x = list(range(1, 11)), height = frequent_itemsets.support[1:11], color =
plt.xticks(list(range(1, 11)), frequent_itemsets.itemsets[1:11], rotation=20)
plt.xlabel('item-sets')
plt.ylabel('support')
plt.show()

rules = association_rules(frequent_itemsets, metric = "lift", min_threshold = 1)
rules
```

```
<ipython-input-4-7144f8a71c82>:8: MatplotlibDeprecationWarning: Using a string
of single character colors as a color sequence is deprecated since 3.2 and will
be removed two minor releases later. Use an explicit list instead.
  plt.bar(x = list(range(1, 11)), height = frequent_itemsets.support[1:11], col
or ='rgmyk')
```




Out[4]:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | levera |
|---|---|---|---|---|---|---|---|---|
| **0** | (ChildBks) | (CookBks) | 0.4230 | 0.431 | 0.2560 | 0.605201 | 1.404179 | 0.0736 |
| **1** | (CookBks) | (ChildBks) | 0.4310 | 0.423 | 0.2560 | 0.593968 | 1.404179 | 0.0736 |
| **2** | (ChildBks) | (GeogBks) | 0.4230 | 0.276 | 0.1950 | 0.460993 | 1.670264 | 0.0782 |
| **3** | (GeogBks) | (ChildBks) | 0.2760 | 0.423 | 0.1950 | 0.706522 | 1.670264 | 0.0782 |
| **4** | (GeogBks) | (CookBks) | 0.2760 | 0.431 | 0.1925 | 0.697464 | 1.618245 | 0.0735 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **289** | (ChildBks, ItalCook) | (GeogBks) | 0.0850 | 0.276 | 0.0525 | 0.617647 | 2.237852 | 0.029( |

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | levera |
|---|---|---|---|---|---|---|---|---|
| **290** | (GeogBks, ItalCook) | (ChildBks) | 0.0640 | 0.423 | 0.0525 | 0.820312 | 1.939273 | 0.0254 |
| **291** | (ChildBks) | (GeogBks, ItalCook) | 0.4230 | 0.064 | 0.0525 | 0.124113 | 1.939273 | 0.0254 |
| **292** | (GeogBks) | (ChildBks, ItalCook) | 0.2760 | 0.085 | 0.0525 | 0.190217 | 2.237852 | 0.0290 |
| **293** | (ItalCook) | (ChildBks, GeogBks) | 0.1135 | 0.195 | 0.0525 | 0.462555 | 2.372077 | 0.0303 |

294 rows × 9 columns

In [5]:
```
rules.head(20)
rules.sort_values('lift', ascending = False).head(10)
```

Out[5]:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage |
|---|---|---|---|---|---|---|---|---|
| **234** | (YouthBks, CookBks) | (ItalCook) | 0.1620 | 0.1135 | 0.0590 | 0.364198 | 3.208789 | 0.040613 |
| **239** | (ItalCook) | (YouthBks, CookBks) | 0.1135 | 0.1620 | 0.0590 | 0.519824 | 3.208789 | 0.040613 |
| **278** | (ItalCook) | (CookBks, ArtBks) | 0.1135 | 0.1670 | 0.0565 | 0.497797 | 2.980822 | 0.037545 |
| **275** | (CookBks, ArtBks) | (ItalCook) | 0.1670 | 0.1135 | 0.0565 | 0.338323 | 2.980822 | 0.037545 |
| **220** | (GeogBks, CookBks) | (ItalCook) | 0.1925 | 0.1135 | 0.0640 | 0.332468 | 2.929229 | 0.042151 |
| **225** | (ItalCook) | (GeogBks, CookBks) | 0.1135 | 0.1925 | 0.0640 | 0.563877 | 2.929229 | 0.042151 |
| **159** | (ItalCook) | (ChildBks, CookBks) | 0.1135 | 0.2560 | 0.0850 | 0.748899 | 2.925385 | 0.055944 |
| **154** | (ChildBks, CookBks) | (ItalCook) | 0.2560 | 0.1135 | 0.0850 | 0.332031 | 2.925385 | 0.055944 |
| **242** | (DoItYBks, CookBks) | (ItalCook) | 0.1875 | 0.1135 | 0.0585 | 0.312000 | 2.748899 | 0.037219 |
| **247** | (ItalCook) | (DoItYBks, CookBks) | 0.1135 | 0.1875 | 0.0585 | 0.515419 | 2.748899 | 0.037219 |

In [6]:
```
################################## Extra part ###################################
#Redudancy is defined as the storing of same data multiple time#
#
def to_list(i):
    return (sorted(list(i)))

ma_X = rules.antecedents.apply(to_list) + rules.consequents.apply(to_list)

ma_X = ma_X.apply(sorted)

rules_sets = list(ma_X)

unique_rules_sets = [list(m) for m in set(tuple(i) for i in rules_sets)]

index_rules = []

for i in unique_rules_sets:
    index_rules.append(rules_sets.index(i))

# getting rules without any redudancy
rules_no_redudancy = rules.iloc[index_rules, :]

# Sorting them with respect to list and getting top 10 rules
rules_no_redudancy.sort_values('lift', ascending = False).head(10)
```

Out[6]:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage |
|---|---|---|---|---|---|---|---|---|
| **234** | (YouthBks, CookBks) | (ItalCook) | 0.1620 | 0.1135 | 0.0590 | 0.364198 | 3.208789 | 0.040613 |
| **220** | (GeogBks, CookBks) | (ItalCook) | 0.1925 | 0.1135 | 0.0640 | 0.332468 | 2.929229 | 0.042151 |
| **154** | (ChildBks, CookBks) | (ItalCook) | 0.2560 | 0.1135 | 0.0850 | 0.332031 | 2.925385 | 0.055944 |
| **242** | (DoItYBks, CookBks) | (ItalCook) | 0.1875 | 0.1135 | 0.0585 | 0.312000 | 2.748899 | 0.037219 |
| **288** | (ChildBks, GeogBks) | (ItalCook) | 0.1950 | 0.1135 | 0.0525 | 0.269231 | 2.372077 | 0.030367 |
| **256** | (DoItYBks, YouthBks) | (RefBks) | 0.1155 | 0.2145 | 0.0580 | 0.502165 | 2.341093 | 0.033225 |
| **62** | (CookBks) | (ItalCook) | 0.4310 | 0.1135 | 0.1135 | 0.263341 | 2.320186 | 0.064582 |
| **268** | (RefBks, ArtBks) | (GeogBks) | 0.0895 | 0.2760 | 0.0565 | 0.631285 | 2.287264 | 0.031798 |
| **138** | (ChildBks, DoItYBks) | (RefBks) | 0.1840 | 0.2145 | 0.0900 | 0.489130 | 2.280328 | 0.050532 |
| **132** | (ChildBks, RefBks) | (GeogBks) | 0.1515 | 0.2760 | 0.0940 | 0.620462 | 2.248051 | 0.052186 |

# Summary:

1- Above the 10 unique Rule that we get by Apply Apriori Algo.

2- Antecedent support variable tells us probability of antecedent product alone.

3- The Support Value is the value of the two Product(Antecedents and Consequents)

4- Confidence is an indication of how often the rule has been found to be True.

5-The ratio of the observed support to that expected if X and Y were independent.

In [ ]:

```

```