Basket Analysis using Association Rules

Market basket analysis is a data mining technique that is used to identify association rules between items in large data sets of transactional data. It is used to identify which items are frequently purchased together in order to understand consumer behavior and to identify potential opportunities for product bundling or cross-selling.

For example, a market basket analysis of a grocery store's transaction data might reveal that customers who purchase bread are very likely to also purchase butter. This information could be used to create a product bundle (e.g., a "baking essentials" bundle including bread and butter) or to cross-sell butter to customers who are purchasing bread (e.g., by placing the butter near the bread in the store).

Benefits of market basket analysis include:

```
Improved customer satisfaction: By understanding which items are frequently purchased together, a retailer can make it e
asier for customers to find everything they need in one place, which can improve the overall shopping experience.

Increased sales: By bundling or cross-selling items that are frequently purchased together, a retailer can increase sale
s of those items.

Enhanced marketing efforts: Market basket analysis can help a retailer understand which items are most likely to be of i
nterest to their customers, which can help them tailor their marketing efforts to better target those customers.

Improved inventory management: By understanding which items are frequently purchased together, a retailer can better pre
dict which items they will need to have in stock at any given time, which can help them manage their inventory more effi
ciently.
```

For this project we will be;

1. Analyse and preprocess the dataset
2. Visualize the weekly, monthly and yearly sales and draw insights from the plotted graphs
3. Visualize the top and bottom selling products
4. Visualize the top customers for this business
5. Genarate association rules to be used to determine the relationships of the products
6. Identify the frequently purchased products

1. Importing the Necessary Libraries

In [1]:
```
##Importing the necessary libraries
import numpy as np
import pandas as pd
import altair as alt
import holoviews as hv

##for visualization styles
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")
plt.style.use('ggplot')
import cufflinks as cf ## mainly used for pandas like visualization, colors, graphs, chart gallery e.t.c
import plotly.express as px
import plotly.offline as py
from plotly.offline import plot
import plotly.graph_objects as go
import plotly.graph_objs as go

## for association rules
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

In [2]:
```
## Reading the dataset
data = pd.read_csv('Groceries_dataset.csv')
data.head()
```

Out[2]:

	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

2. Data Preprocessing

In [3]:
```
data.shape
```

Out[3]:
```
(38765, 3)
```

In [4]:
```
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
```

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

In [6]:
```
## Number of unique items in the dataset
print(f'There are : {len(data["itemDescription"].unique())} unique items')
```

```
There are : 167 unique items
```

In [7]:
```
##Renaming the columns
data.rename(columns = {'Member_number':'id','itemDescription':'item'}, inplace = True)
```

In [8]:
```
##Splitting the date column to form new separate columns for the day, month and year
#Convert the 'Date' column to datetime format
data['Date']= pd.to_datetime(data['Date'])

#Extracting year,month and day
data['year'] = data['Date'].apply(lambda x : x.year)
data['month'] = data['Date'].apply(lambda x : x.month)
data['day'] = data['Date'].apply(lambda x : x.day)
data['weekday'] = data['Date'].apply(lambda x : x.weekday())

#Rearranging the columns
dataFrame=data[['id', 'Date','year', 'month', 'day','weekday','item']]
dataFrame.head()
```

Out[8]:

	id	Date	year	month	day	weekday	item
0	1808	2015-07-21	2015	7	21	1	tropical fruit
1	2552	2015-05-01	2015	5	1	4	whole milk
2	2300	2015-09-19	2015	9	19	5	pip fruit
3	1187	2015-12-12	2015	12	12	5	other vegetables
4	3037	2015-01-02	2015	1	2	4	whole milk

In [9]:
```
dataFrame["year"].unique()
```

Out[9]:
```
array([2015, 2014], dtype=int64)
```

In [10]:
```
dataFrame["month"].unique()
```

Out[10]:
```
array([ 7,  5,  9, 12,  1,  2,  8,  3,  4, 11, 10,  6], dtype=int64)
```

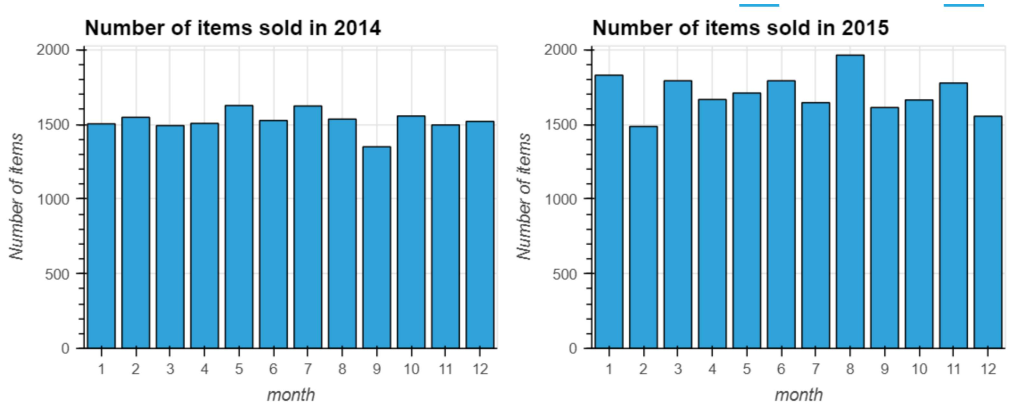
3. Explaratory Data Analysis

1. The business yearly sales

In [11]:
```
from holoviews import opts
hv.extension('bokeh')

#Filter data for 2014 and 2015
df1=dataFrame.groupby(['year']).filter(lambda x: (x['year'] == 2014).any())
df2=dataFrame.groupby(['year']).filter(lambda x: (x['year'] == 2015).any())

#Monthly data for the two years
sales_2014=hv.Bars(df1.groupby(['month'])['item'].count()).opts(ylabel="Number of items", title='Number of items sold in 2014')
sales_2015=hv.Bars(df2.groupby(['month'])['item'].count()).opts(ylabel="Number of items", title='Number of items sold in 2015')

#Combining the two plots
(sales_2014 + sales_2015).opts(opts.Bars(width=380, height=300,tools=['hover'],show_grid=True))
```

Out[11]:

(https://bokeh.org/)

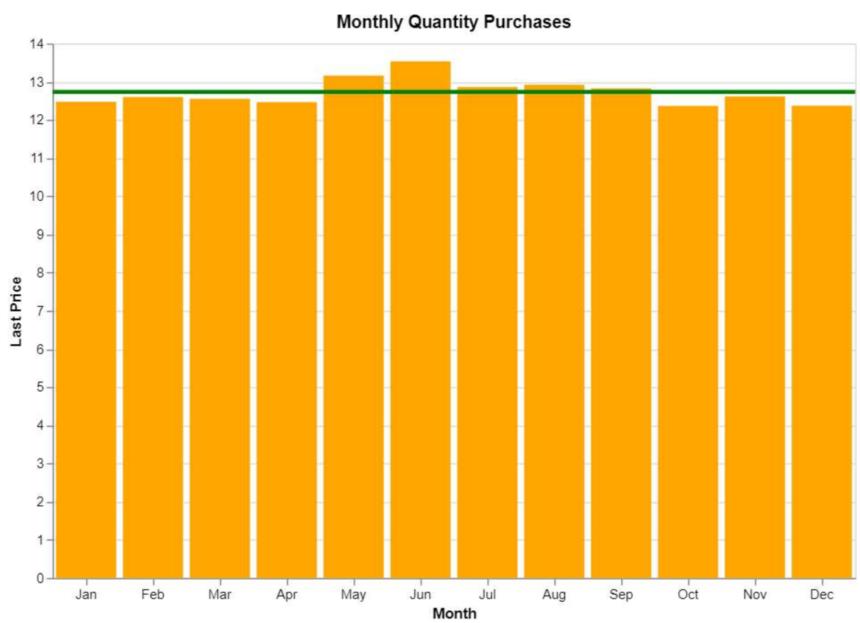


1. The year 2015 has a the higher sales than 2014 thus the business has a progressive increase in sales between the two years
2. October 2015 has the highest sales
3. In February and September recorded the lowest sales in 2015 and september 2014 also has the lowest sales

2. The business monthly quantity purchases

In [12]:
```
#Creating temporary data which has quantity purchased column
temp=dataFrame.copy()
temp['qty_purchased']=dataFrame['id'].map(dataFrame['id'].value_counts())

#Slicing first 5000 rows as altair library can't plot any data which has record beyond that
temp1=temp[:5000]
temp1.columns

#Plotting
brush = alt.selection(type='interval', encodings=['x'])

#Plotting the bar chart
bars = alt.Chart().mark_bar(color="orange").encode(
    x=alt.X('month(Date):O',title="Month"),
    y=alt.Y('mean(qty_purchased):Q',title="Last Price"),
    opacity=alt.condition(brush, alt.OpacityValue(1), alt.OpacityValue(0.7)),
    tooltip=['month(Date)','mean(qty_purchased)']
).add_selection(
    brush
).properties(height=400,width=600,title="Monthly Quantity Purchases")

#Plotting avrage line
line = alt.Chart().mark_rule(color='green').encode(
    y='mean(qty_purchased):Q',
    size=alt.SizeValue(3),
    tooltip=['mean(qty_purchased)']
).transform_filter(
    brush
)

#Display plot using sliced data
alt.layer(bars, line, data=temp1)
```

Out[12]:



1. The top difference was experienced when the band shift after Jan-Apr months(window-4 months)
2. The highest average is obviously between May-Aug month where highest was from June

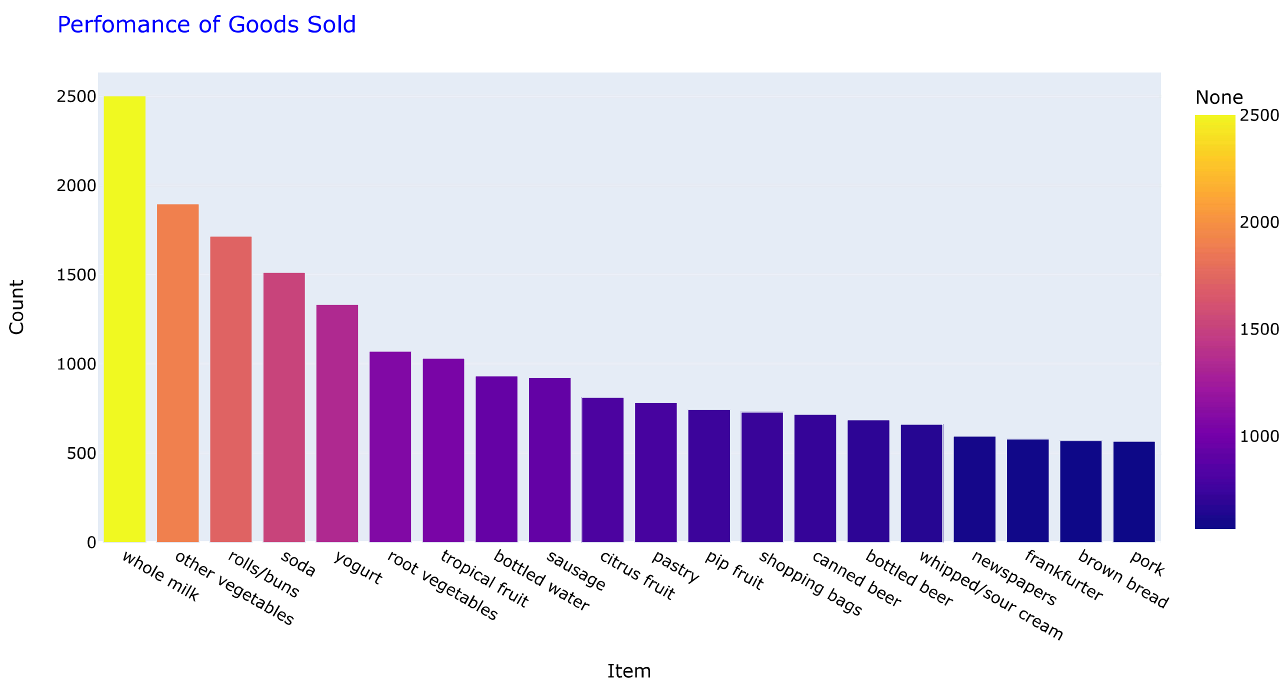
3. The business weekly quantity purchases

In [13]:
```
#Converting weekday variable to category
temp1.weekday = temp1.weekday.astype('category')

#Creating a new dataframe which has the frequency of weekdays
weekday_bin=temp1['weekday'].value_counts().to_frame().reset_index().rename(columns={'index':'weekday','weekday':'count'})

#Plotting bar chart
bars = alt.Chart(weekday_bin).mark_bar(color="darkgreen").encode(
    x='weekday',
    y=alt.Y("count",title='Number of purchases')
)

#Adding data labels
text = bars.mark_text(
    align='center',
    baseline='middle',
    dy=-7 ,
    size=15,
).encode(
    text='count',
    tooltip=[alt.Tooltip('weekday'),
            alt.Tooltip('count')]
)

#Combining both
(bars + text).properties(
    width=800,
    height=400,
    title="Weekly Quantity Purchases"
)
```

Out[13]:

4. Perfomance of the products

In [14]:
```
#Graph : Item by count
fig = px.bar(dataFrame["item"].value_counts()[:20], orientation="v", color=dataFrame["item"].value_counts()[:20], color_continuou
             log_x=False, labels={'value':'Count',
                                   'index':'Item',
                                    'color':'None'
                                   })

fig.update_layout(
    font_color="black",
    title_font_color="blue",
    legend_title_font_color="green",
    title_text="Perfomance of Goods Sold"
)

fig.show()
```

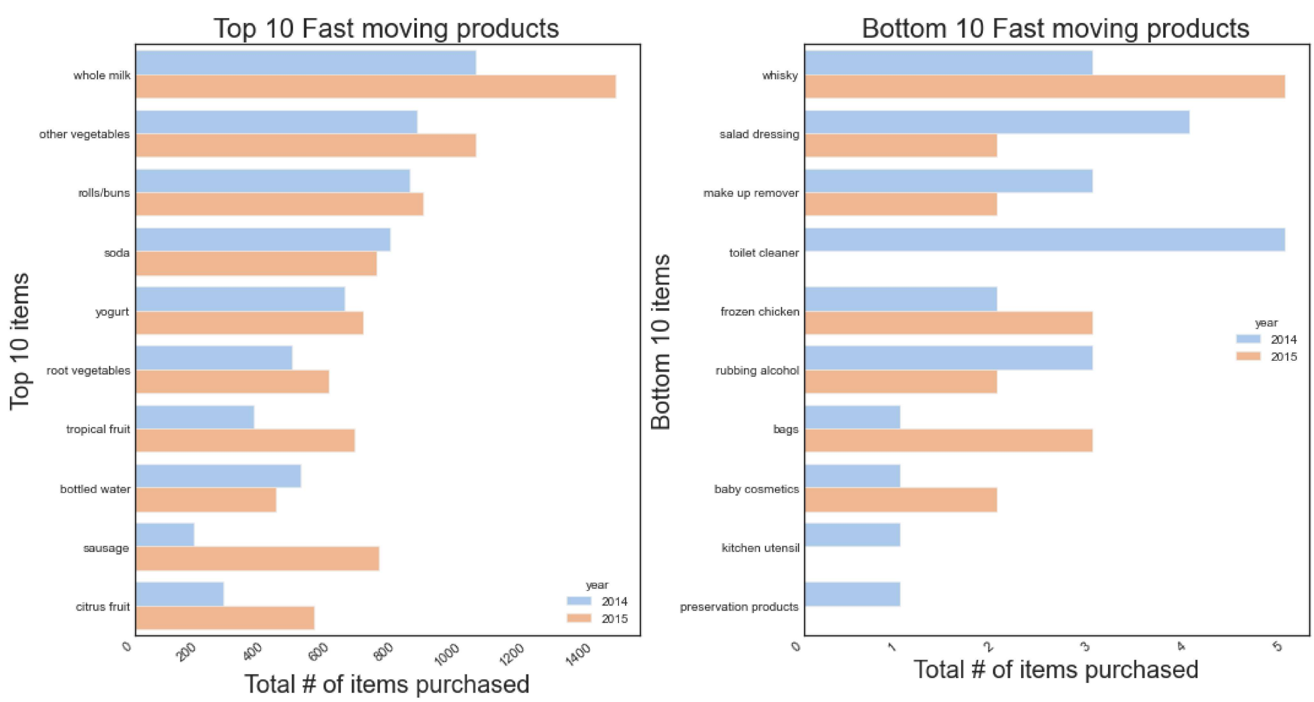


1. Whole milk products have the highest sales
2. Pork is the least purchased product
3. The top 10 purchased product include food items

5. The top and bottom 10 Fast moving products in both years

In [15]:
```
#Setting plot style
plt.figure(figsize = (15, 8))
plt.style.use('seaborn-white')

#Top 10 fast moving products
plt.subplot(1,2,1)
ax=sns.countplot(y="item", hue="year", data=dataFrame, palette="pastel",
              order=data.item.value_counts().iloc[:10].index)

ax.set_xticklabels(ax.get_xticklabels(),fontsize=11,rotation=40, ha="right")
ax.set_title('Top 10 Fast moving products',fontsize= 22)
ax.set_xlabel('Total # of items purchased',fontsize = 20)
ax.set_ylabel('Top 10 items', fontsize = 20)
plt.tight_layout()

#Bottom 10 fast moving products
plt.subplot(1,2,2)
ax=sns.countplot(y="item", hue="year", data=dataFrame, palette="pastel",
              order=data.item.value_counts().iloc[-10:].index)
ax.set_xticklabels(ax.get_xticklabels(),fontsize=11,rotation=40, ha="right")
ax.set_title('Bottom 10 Fast moving products',fontsize= 22)
ax.set_xlabel('Total # of items purchased',fontsize = 20)
ax.set_ylabel('Bottom 10 items', fontsize = 20)
plt.tight_layout()
```

```
C:\Users\Lian.s\AppData\Local\Temp/ipykernel_13552/919842071.py:3: MatplotlibDeprecationWarning:

The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seabor
n. However, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.

C:\Users\Lian.s\AppData\Local\Temp/ipykernel_13552/919842071.py:10: UserWarning:

FixedFormatter should only be used together with FixedLocator

C:\Users\Lian.s\AppData\Local\Temp/ipykernel_13552/919842071.py:20: UserWarning:

FixedFormatter should only be used together with FixedLocator
```

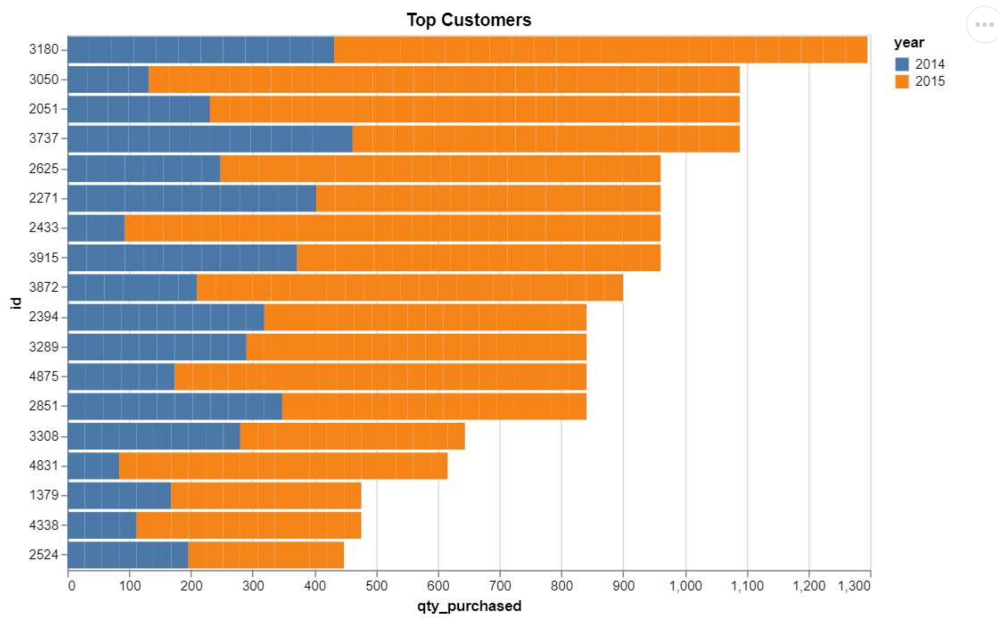


1. Milk is the top product purchased in both 2014 and 2015 whereas lowest is preservation product which no one purchased in 2015
2. Almost all the top products has seen a rise in 2015 except soda and bottled water
3. Most of the bottom products nevr saw a rise in 2015 except whiskey,chicken,bag and baby cosmetics

6.The top customers with most purchases in both years

In [16]:
```
top_customers=temp[['id', 'qty_purchased','year']].sort_values(by = 'qty_purchased',ascending = False).head(500)
top_customers.id = top_customers.id.astype('category')
top_customers.year = top_customers.year.astype('category')
alt.Chart(top_customers).mark_bar(color="blue").encode(
    x='qty_purchased',
    y=alt.Y('id', sort='-x'),
    color='year',
    tooltip=['id','qty_purchased']
).properties(height=400,width=600,title="Top Customers")
```

Out[16]:



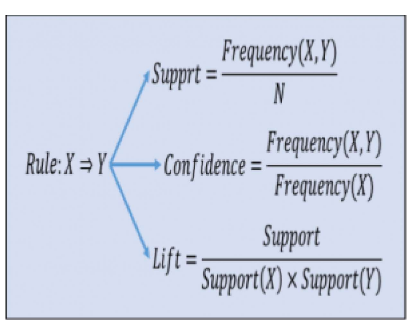
1. 3180 id customer has topped the list and has been a loyal customer in both the year
2. There can be few customers who are seen to be inconsitent where they have purchased a lot in 2014 and not in 2015 when it comes to customer life expectancy these consistency are considered. Since we have only two year data we can't comment on each customer about their customer life expectancy much

4. Association Rules with Apriori Algorithm

Association Rules are based on the concept of strong rules, are widely used to analyze retail basket or transaction data and are intended to identify strong rules discovered in transaction data using measures of interestingness. Association rules are used to unveil the relationship between one item and another when purchased, mainly denotes as X and Y. X is the main product being purchased while Y is the best product to be bought together with X. These rules are developed by three terminologies;

```
1. Support - It is used to represent the number of transactions in which product X appears from thee total number of tra
nsactions. That is, the popularity of product X.
2. Confidence - It is the likelyhood that product Y being purchased when item X is purchased.
3. Lift -  This says how likely item Y is purchased when item X is purchased while controlling for how popular item Y i
s.
```

They are calculated as follows;



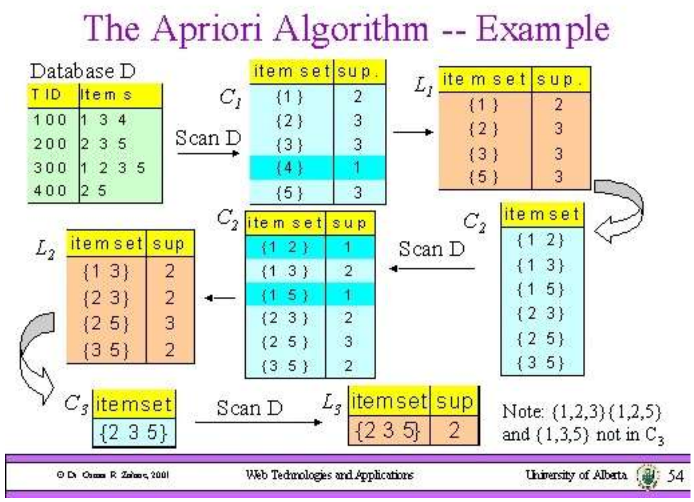
The Apriori algorithm generates association rules, but it does so under the condition that

```
1. All subsets of the frequent itemset must all be frequent.
2. For any infrequent itemset all it's supersets must be infrequent too
```

The method may take some time to construct if all rules are taken into account, thus if the lift of these chosen itemsets (rules) is less than a threshold, the rules are removed. the Apriori algorithm generates association rules, but it does so under the condition that

```
1. Subsets of the frequent itemset must all be frequent.
2. Similar to this, the algorithm operates in such a way that iterations take place with frequent itemsets and a minimum
support value is determined if an infrequent subset has an infrequent parent set. Until removal is impossible, itemsets
and subsets are disregarded if their support falls below the threshold.
```

The method may take some time to construct if all rules are taken into account, thus if the lift of these chosen itemsets (rules) is less than a threshold, the rules are removed.



Apriori Algorithm Steps

1. Identify the support threshold, in the above example, the support threshold has been set to 1
2. Eliminate the items in the dataset that do not meet the requirements of the support threshold
3. Pair up the items into 2 itemsets in the dataset
4. From the pairs formed, eliminate the items with support of 1 and below
5. Form pairs of three with the remaining items in the dataset
6. Remove the items below the threshold
7. The remaining pair is the valid frequent items purchased together

4.1 Preparing items for the Apriori Algorithm

In [17]:
```
from apyori import apriori
```

In [19]:
```
## Grouping the member's basket by the id, date and the number of items in the basket
transactions = basket.groupby(['Member_number','Date'])
transactions.count()
```

Out[19]:

		itemDescription
Member_number	**Date**	
1000	**15-03-2015**	4
	24-06-2014	3
	24-07-2015	2
	25-11-2015	2
	27-05-2015	2
...	**...**	...
4999	**24-01-2015**	6
	26-12-2015	2
5000	**09-03-2014**	2
	10-02-2015	3
	16-11-2014	2

14963 rows × 1 columns

In [20]:
```
## Let's try to see what items where in the baskets from transactions
list_transactions = [i[1]['itemDescription'].tolist() for i in list(transactions)]
list_transactions[:20]
```

Out[20]:
```
[['sausage', 'whole milk', 'semi-finished bread', 'yogurt'],
 ['whole milk', 'pastry', 'salty snack'],
 ['canned beer', 'misc. beverages'],
 ['sausage', 'hygiene articles'],
 ['soda', 'pickled vegetables'],
 ['frankfurter', 'curd'],
 ['sausage', 'whole milk', 'rolls/buns'],
 ['whole milk', 'soda'],
 ['beef', 'white bread'],
 ['frankfurter', 'soda', 'whipped/sour cream'],
 ['frozen vegetables', 'other vegetables'],
 ['butter', 'whole milk'],
 ['tropical fruit', 'sugar'],
 ['butter milk', 'specialty chocolate'],
 ['sausage', 'rolls/buns'],
 ['root vegetables', 'detergent'],
 ['frozen meals', 'dental care'],
 ['rolls/buns', 'rolls/buns'],
 ['dish cleaner', 'cling film/bags'],
 ['canned beer', 'frozen fish']]
```

In [21]:
```
## If we combine the two lists together, we can see the items in the customer's basket
## Grouping the member's basket by the id, date and the number of items in the basket
transactions = basket.groupby(['Member_number','Date','itemDescription'])
transactions.count()
```

Out[21]:

Member_number	Date	itemDescription
1000	**15-03-2015**	**sausage**
		semi-finished bread
		whole milk
		yogurt
	24-06-2014	**pastry**
...	**...**	**...**
5000	**10-02-2015**	**root vegetables**
		semi-finished bread
		soda
	16-11-2014	**bottled beer**
		other vegetables

38006 rows × 0 columns

We can see the 4 items in customer 1000, bought on the 15th of March were 'sausage', 'whole milk', 'semi-finished bread', 'yogurt'

4.2 Building Apriori Algorithm from the customer's baskets

In [22]:
```
## Building the rules for apriori with the customer's baskets in list_transactions
rules = apriori(list_transactions, min_support=0.001, min_confidence=0.05, min_lift=1.2, min_length=2, max_length=2)
```

In [23]:
```
results = list(rules)
```

In [24]:
```
def inspect(results):
    lhs = [tuple(result[2][0][0])[0] for result in results]
    rhs = [tuple(result[2][0][1])[0] for result in results]
    support = [result[1]*100 for result in results]
    confidence = [result[2][0][2]*100 for result in results]
    lift = [result[2][0][3] for result in results]
    return list(zip(lhs,rhs,support,confidence,lift))
final_result = pd.DataFrame(inspect(results), columns=['Antecedent','Consequent','Support(%)','Confidence(%)','lift'])
final_result['Rule'] = final_result['Antecedent'] + '->' + final_result['Consequent']
```

In [25]:
```
final_result
```

Out[25]:

	Antecedent	Consequent	Support(%)	Confidence(%)	lift	Rule
0	beverages	sausage	0.153712	9.274194	1.536764	beverages->sausage
1	bottled beer	sausage	0.334158	7.374631	1.222000	bottled beer->sausage
2	sugar	bottled water	0.147029	8.301887	1.368074	sugar->bottled water
3	brown bread	canned beer	0.240593	6.394316	1.362937	brown bread->canned beer
4	candy	citrus fruit	0.100247	6.976744	1.313120	candy->citrus fruit
5	white bread	canned beer	0.153712	6.406685	1.365573	white bread->canned beer
6	cat food	tropical fruit	0.100247	8.474576	1.250543	cat food->tropical fruit
7	chewing gum	yogurt	0.140346	11.666667	1.358508	chewing gum->yogurt
8	specialty chocolate	citrus fruit	0.140346	8.786611	1.653762	specialty chocolate->citrus fruit
9	curd	sausage	0.294059	8.730159	1.446615	curd->sausage
10	detergent	yogurt	0.106930	12.403101	1.444261	detergent->yogurt
11	flour	tropical fruit	0.106930	10.958904	1.617141	flour->tropical fruit
12	frozen meals	sausage	0.126980	7.569721	1.254327	frozen meals->sausage
13	frozen vegetables	sausage	0.207178	7.398568	1.225966	frozen vegetables->sausage
14	grapes	sausage	0.106930	7.407407	1.227431	grapes->sausage
15	hard cheese	pip fruit	0.106930	7.272727	1.482586	hard cheese->pip fruit
16	herbs	yogurt	0.113614	10.759494	1.252874	herbs->yogurt
17	napkins	pastry	0.173762	7.854985	1.518529	napkins->pastry
18	oil	soda	0.180445	12.107623	1.246844	oil->soda
19	onions	whipped/sour cream	0.106930	5.280528	1.208143	onions->whipped/sour cream
20	packaged fruit/vegetables	rolls/buns	0.120297	14.173228	1.288421	packaged fruit/vegetables->rolls/buns
21	processed cheese	rolls/buns	0.147029	14.473684	1.315734	processed cheese->rolls/buns
22	processed cheese	root vegetables	0.106930	10.526316	1.513019	processed cheese->root vegetables
23	seasonal products	rolls/buns	0.100247	14.150943	1.286395	seasonal products->rolls/buns
24	sliced cheese	root vegetables	0.120297	8.571429	1.232030	sliced cheese->root vegetables
25	sliced cheese	sausage	0.113614	8.095238	1.341407	sliced cheese->sausage
26	soft cheese	yogurt	0.126980	12.666667	1.474952	soft cheese->yogurt
27	specialty chocolate	tropical fruit	0.133663	8.368201	1.234846	specialty chocolate->tropical fruit

In [ ]: