Market Basket Insights

PROMBLEM DEFINATION :-

- Market basket analysis technique used by retailers to increase sales by better understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase history, to reveal product groupings, as well as products that are likely to be purchased together
- Compared to handwritten records kept by store owners, the digital records generated by POS systems made it easier for applications to process analyze large volume of purchase

DESIGN THINKING:-

- In market basket analysis,rules are used to predict the likelihood of products being purchased together. Association rules count the frequency of items that occur together, seeking to find associations that occur far more often than expected.
- Amazon's website uses a well-known example of market basket analysis. On a product page, Amazon presents users with related products, under the headings of Frequently bought together" and "Customers who bought this item also bought.

- Market basket analysis also applies to bricks-and-mortar stores. If analysis showed that magazine purchases often include the purchase of a bookmark, which could be considered an unexpected combination as the consumer did not purchase a book, then the bookstore might place a selection of bookmarks near the magazine rack.
- Algorithms that use association rules include AIS, SETM and Apriori. The Apriori algorithm is commonly cited by data scientists in research articles about market basket
- The Arules package for R is an open source toolkit for association mining using the R programming language. This package supports the Apriori algorithm, along with the following other mining algorithms

DEVELOPMENT PART:

Hi! In this kernel we are going to use the Apriori algorithm to perform a Market Basket Analysis. A Market what? Is a technique used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions, providing information to understand the purchase behavior. The outcome of this type of technique is, in simple terms, a set of rules that can be understood as "if this, then that". For more information about these topics, please check in the following links:

1. Association rules

The Apriori algorithm generates association rules for a given data set. An association rule implies that if an item A occurs, then item B also occurs with a certain probability. Let's see an example,

In the table above we can see seven transactions from a clothing store. Each transaction

Transaction	Items
t1	{T-shirt, Trousers, Belt}
t2	{T-shirt, Jacket}
t3	{T-shirt, Trousers, Jacket}
t4	{T-shirt, Trousers, Sneakers, Jacket, Belt}
t5	{Trousers, Sneakers, Belt}
t6	{Trousers, Belt, Sneakers}

shows items bought in that transaction. We can represent our items as an item set as follows:

$$I=\{i1,i2,...,ik\}=\{1,2,...,\}$$

In our case it corresponds to:

$$I=\{T\text{-shirt},Trousers,Belt,Jacket,Gloves,Sneakers\}=\{\text{-}\hbar\ ,,,,,\}$$

A transaction is represented by the following expression:

$$T=\{t1,t2,...,tn\}=\{1,2,...,\}$$

For example,

$$t1=\{T\text{-shirt,Trousers,Belt}\}1=\{\text{-}\hbar\ ,,\}$$

Then, an association rule is defined as an implication of the form:

$$X\Rightarrow Y\Rightarrow, \text{ where } X\subset I\subset, Y\subset I\subset \text{ and } X\cap Y=0\cap=0$$

For example,

$$\{T\text{-shirt,Trousers}\}\Rightarrow\{Belt\}\{\text{-}\hbar\ ,\}\Rightarrow\{\}$$

In the following sections we are going to define four metrics to measure the precision of a rule.

2. Loading Data

First we need to load some libraries and import our data. We can use the function read.transactions() from the arules package to create a transactions object.

Code

```
# Load libraries

library(tidyverse) # data manipulation

library(arules) # mining association rules and frequent itemsets
library(arulesViz) # visualization techniques for association rules
library(knitr) # dynamic report generation

library(gridExtra) # provides a number of user-level functions to work with "grid"
graphics library(lubridate) # work with dates and times

# Read the data

trans <- read.transactions("../input/BreadBasket_DMS.csv",
format="single", cols=c(3,4), sep=",", rm.duplicates=TRUE)
```

3. Data Dictionary

The data set contains 15.010 observations and the following columns,

Date. Categorical variable that tells us the date of the transactions (YYYY-MM-DD format). The column includes dates from 30/10/2016 to 09/04/2017.

Time. Categorical variable that tells us the time of the transactions (HH:MM:SS format).

Transaction. Quantitative variable that allows us to differentiate the transactions. The rows that share the same value in this field belong to the same transaction, that's why the data set has less transactions than observations.

Item. Categorical variable containing the products.

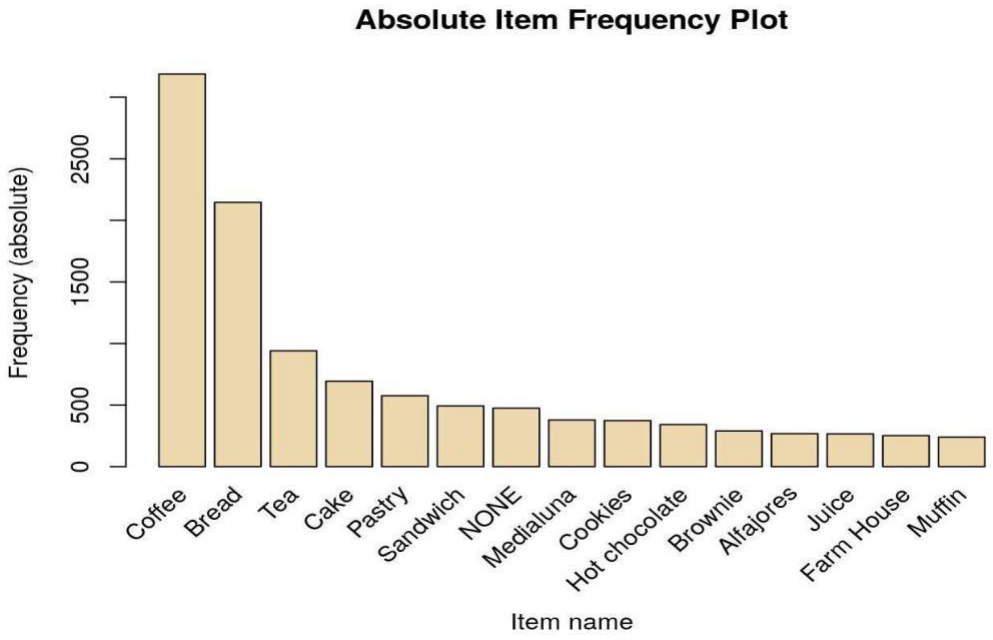
4. Data Analysis

Before applying the Apriori algorithm on the data set, we are going to show some visualizations to learn more about the transactions. For example, we can use the itemFrequencyPlot() function to create an item frequency bar plot, in order to view the distribution of products.

Code

```
# Absolute Item Frequency Plot

itemFrequencyPlot(trans, topN=15, type="absolute", col="wheat2",xlab="Item name", ylab="Frequency (absolute)", main="Absolute Item Frequency Plot")
```

OUTPUT:

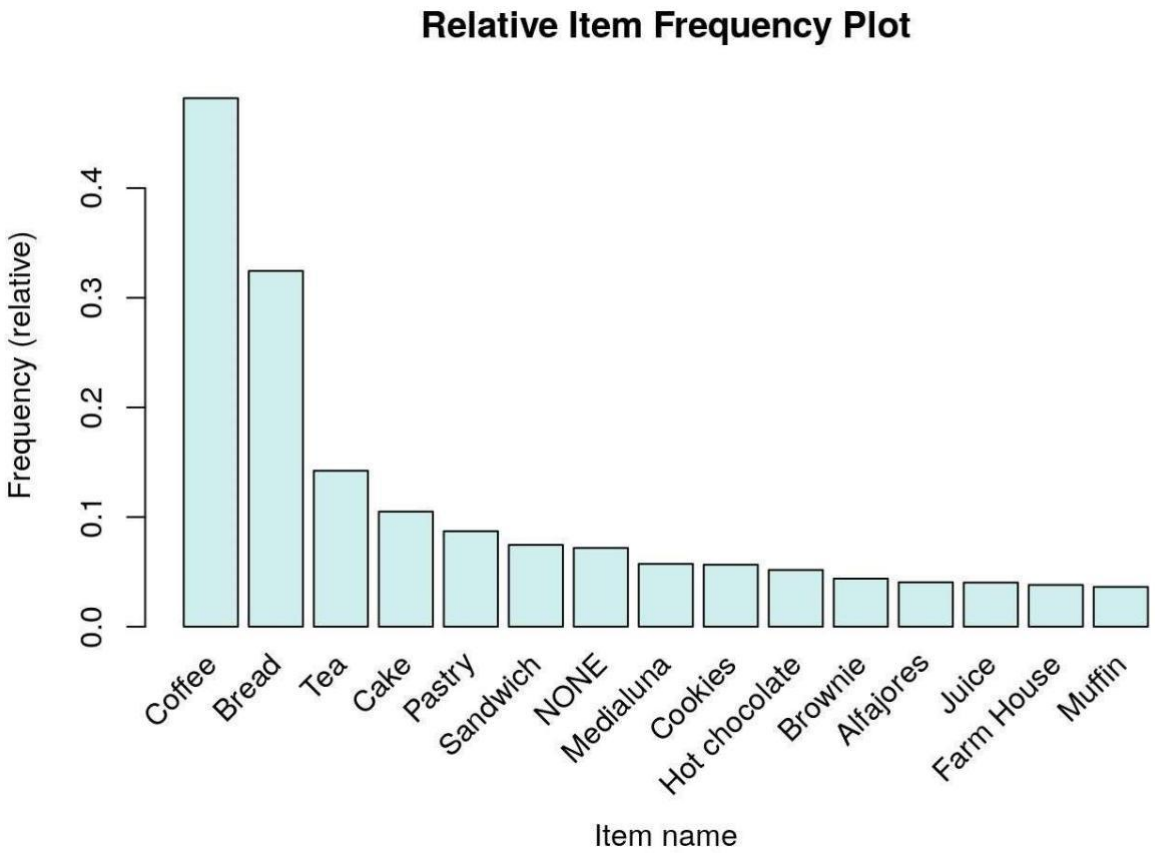


The itemFrequencyPlot() allows us to show the absolute or relative values. If absolute it will plot numeric frequencies of each item independently. If relative it

will plot how many times these items have appeared as compared to others, as it's shown in the following plot.

Code

```
# Relative Item Frequency Plot
itemFrequencyPlot(trans, topN=15, type="relative", col="lightcyan2", xlab="Item name",
ylab="Frequency (relative)", main="Relative Item Frequency Plot")
```



OUTPUT:

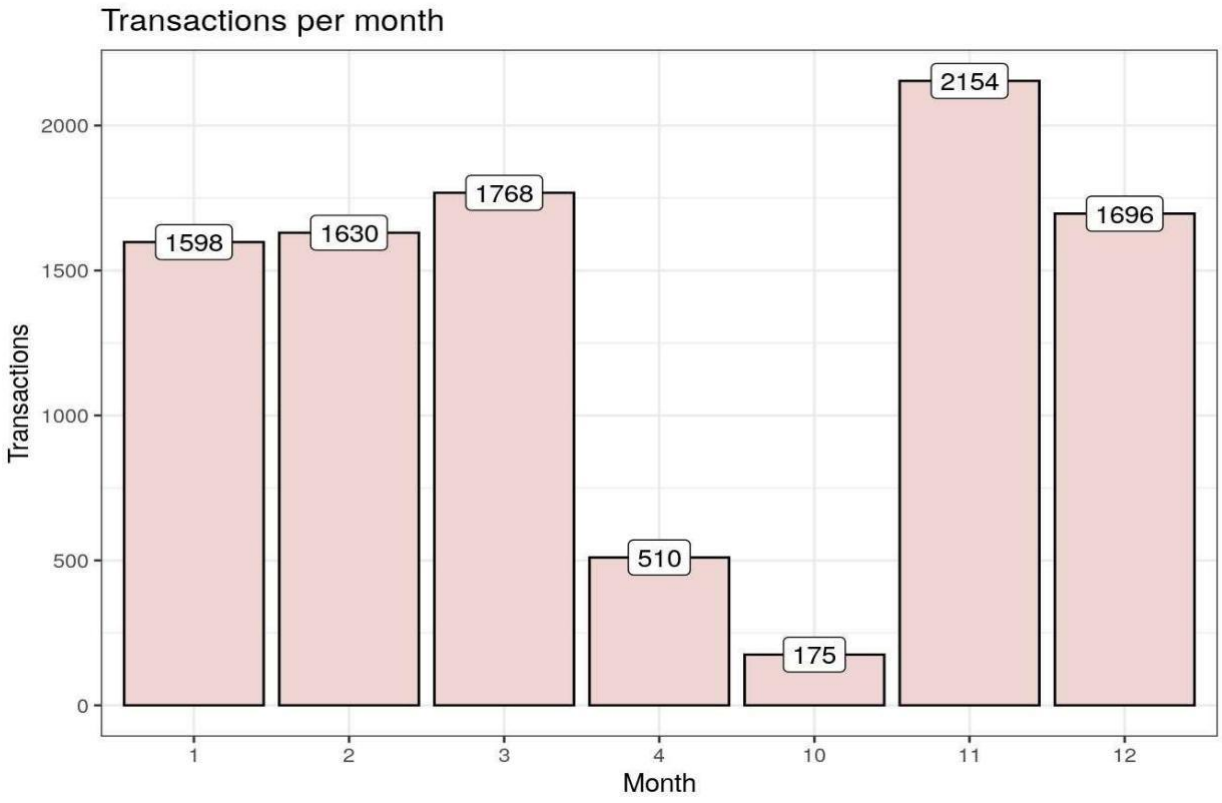
Coffee is the best-selling product by far, followed by bread and tea. Let's display some other visualizations describing the time distribution using the ggplot() function.

Code

```
# Load data
```

```
trans_csv <-read.csv("../input/BreadBasket_DMS.csv")

# Visualization - Transactions per
 month trans_csv%>%
mutate(Month=as.factor(month(Date))) %>%
 group_by(Month) %>%
summarise(Transactions=n_distinct(Transaction)) %>%
ggplot(aes(x=Month, y=Transactions)) +
geom_bar(stat="identity", fill="mistyrose2",
show.legend=FALSE, colour="black") +
geom_label(aes(label=Transactions)) + labs(title="Transactions
per month") + theme_bw()
```

OUTPUT:

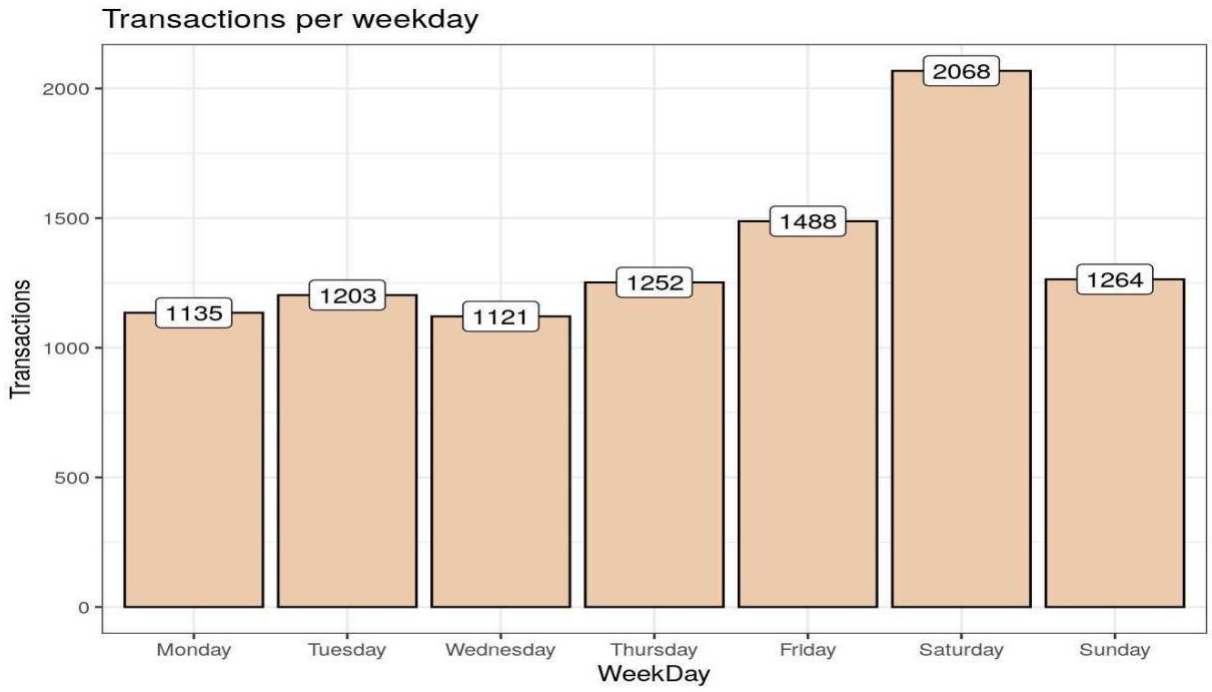


The data set includes dates from 30/10/2016 to 09/04/2017, that's why we have so few transactions in October and April.

Code

```
# Visualization - Transactions per weekday

trans_csv %>%
mutate(WeekDay=as.factor(weekdays(as.Date(Date)))) %>%
group_by(WeekDay) %>%
summarise(Transactions=n_distinct(Transaction)) %>%
ggplot(aes(x=WeekDay, y=Transactions)) +

geom_bar(stat="identity", fill="peachpuff2",

 show.legend=FALSE, colour="black") +

geom_label(aes(label=Transactions)) +

 labs(title="Transactions per weekday") +

scale_x_discrete(limits=c("Monday", "Tuesday", "Wednesday", "Thursday",

"Friday", "Saturday", "Sunday")) +theme_bw()
```

OUTPUT:

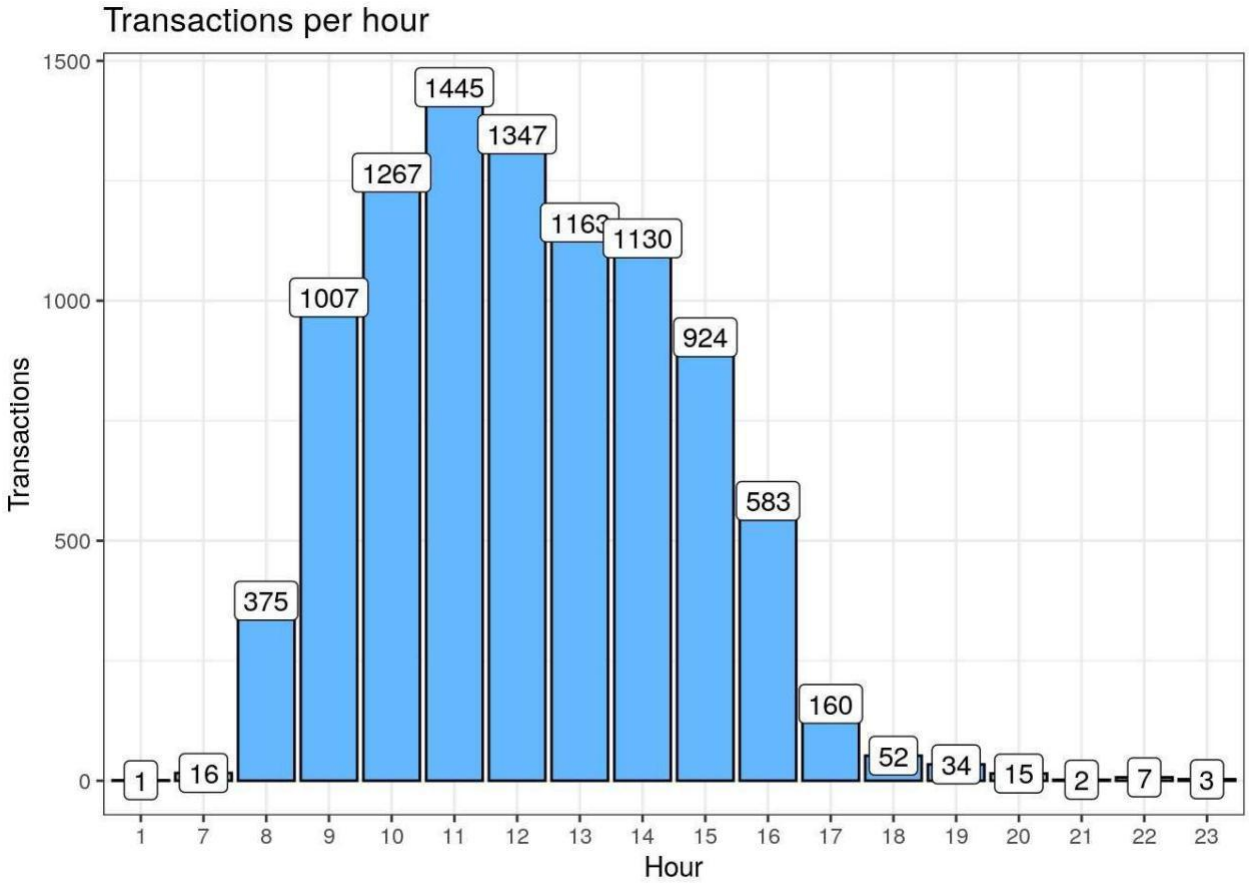


As we can see, Saturday is the busiest day in the bakery. Conversely, Wednesday is the day with fewer transactions.

Code:

```
# Visualization - Transactions per hour
trans_csv %>%
mutate(Hour=as.factor(hour(hms(Time))))%>%
group_by(Hour) %>%
summarise(Transactions=n_distinct(Transaction))
%>% ggplot(aes(x=Hour, y=Transactions)) +
geom_bar(stat="identity", fill="steelblue1", show.legend=FALSE,
colour="black") + geom_label(aes(label=Transactions)) +
labs(title="Transactions per
hour") + theme_bw()
```

OUTPUT:



There's not much to discuss with this visualization. The results are logical and expected.

5. Apriori algorithm

A) Choice of support and confidence

The first step in order to create a set of association rules is to determine the optimal thresholds for support and confidence. If we set these values too low, then the algorithm will take longer to execute and we will get a lot of rules (most of them will not be useful). Then, what values do we choose? We can try different values of support and confidence and see graphically how many rules are generated for each combination.

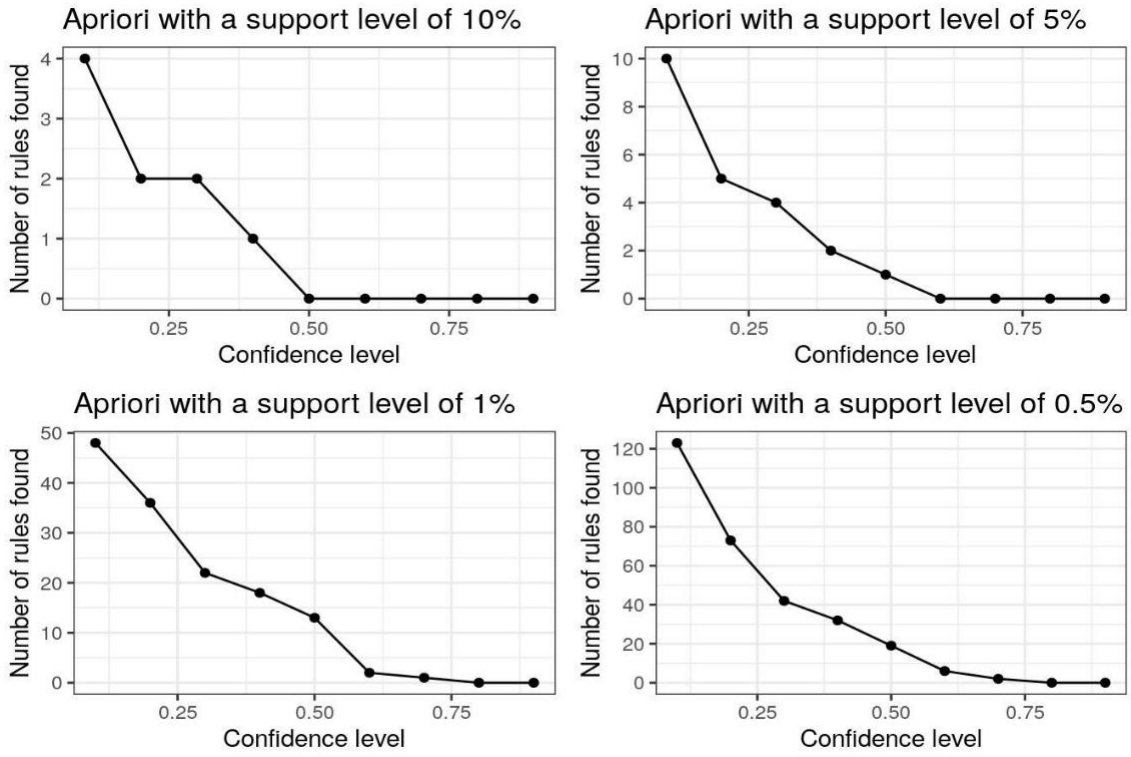
Code

```
# Support and confidence values

supportLevels <- c(0.1, 0.05, 0.01, 0.005)

confidenceLevels <- c(0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1)
# Empty integers rules_sup10

<- integer(length=9) rules_sup5

<- integer(length=9) rules_sup1

<- integer(length=9)

rules_sup0.5 <-

integer(length=9)

# Apriori algorithm with a support level of

10% for (i in 1:length(confidenceLevels)) {

  rules_sup10[i] <- length(apriori(trans, parameter=list(sup=supportLevels[1],

                          conf=confidenceLevels[i], target="rules")))
}
# Apriori algorithm with a support level of 5%

for (i in 1:length(confidenceLevels)){
  rules_sup5[i] <- length(apriori(trans, parameter=list(sup=supportLevels[2],
  conf=confidenceLevels[i], target="rules")))
}
# Apriori algorithm with a support level of

1% for (i in 1:length(confidenceLevels)){
```

```
  rules_sup1[i] <- length(apriori(trans, parameter=list(sup=supportLevels[3],
                          conf=confidenceLevels[i], target="rules")))
}
# Apriori algorithm with a support level of 0.5%
for (i in 1:length(confidenceLevels)){
  rules_sup0.5[i] <- length(apriori(trans, parameter=list(sup=supportLevels[4],
                          conf=confidenceLevels[i], target="rules")))
}
```

In the following graphs we can see the number of rules generated with a support level of 10%,
5%, 1% and 0.5%.Code

OUTPUT;



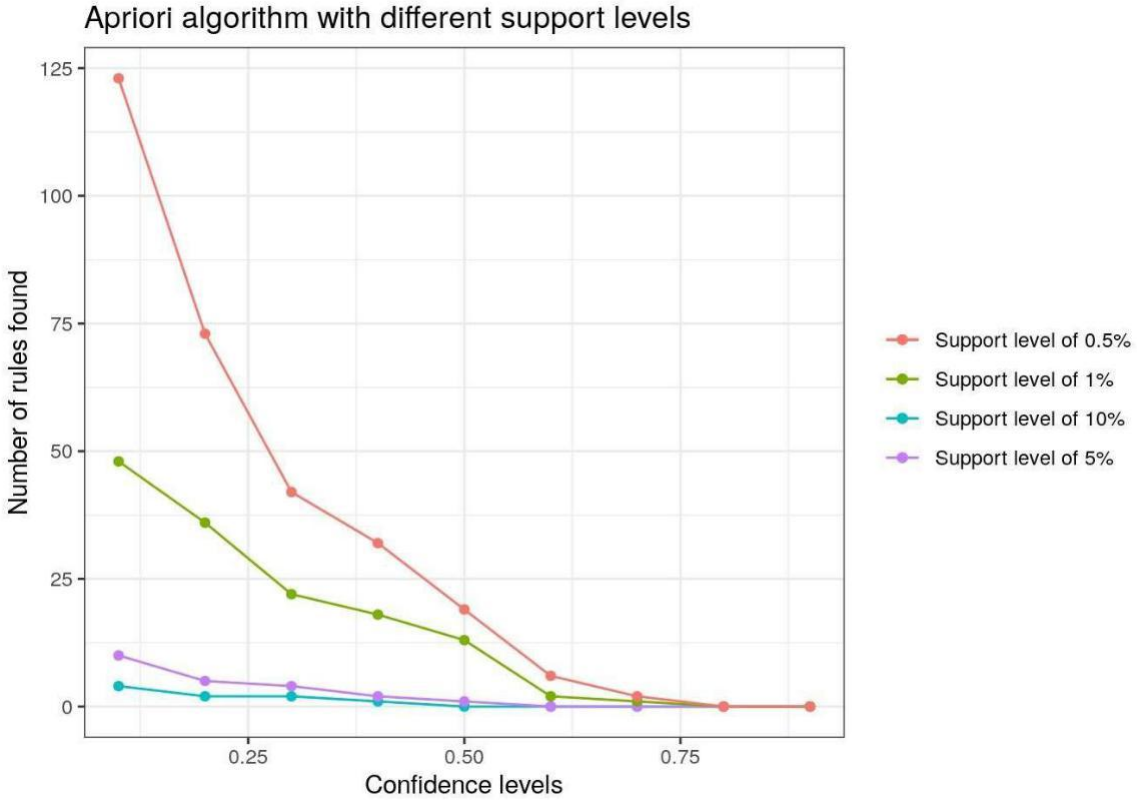
We can join the four lines to improve the visualization.

Code

```
# Data frame

num_rules <- data.frame(rules_sup10, rules_sup5, rules_sup1,
rules_sup0.5, confidenceLevels)
```

```
# Number of rules found with a support level of 10%, 5%, 1% and
0.5% ggplot(data=num_rules, aes(x=confidenceLevels)) +


# Plot line and points (support level of 10%)
  geom_line(aes(y=rules_sup10, colour="Support level of
  10%")) + geom_point(aes(y=rules_sup10, colour="Support
  level of 10%")) + # Plot line and points (support level of 5%)
  geom_line(aes(y=rules_sup5, colour="Support level of
  5%")) + geom_point(aes(y=rules_sup5, colour="Support
  level of 5%")) + # Plot line and points (support level of 1%)
  geom_line(aes(y=rules_sup1, colour="Support level of 1%"))
  + geom_point(aes(y=rules_sup1, colour="Support level of
  1%")) + # Plot line and points (support level of 0.5%)
  geom_line(aes(y=rules_sup0.5, colour="Support level of
  0.5%")) + geom_point(aes(y=rules_sup0.5,
  colour="Support level of 0.5%")) + # Labs and theme
  labs(x="Confidence levels", y="Number of rules found",

title="Apriori algorithm with different support levels") +theme_bw()
+theme(legend.title=element_blank())
```

OUTPUT:



Let's analyze the results,

B) Execution

Let's execute the Apriori algorithm with the values obtained in the previous section.

Code

```
# Apriori algorithm execution with a support level of 1% and a confidence level of
50% rules_sup1_conf50 <- apriori(trans, parameter=list(sup=supportLevels[3],
                    conf=confidenceLevels[5], target="rules"))
```

The generated association rules are the following,

Code

```
# Inspect association rules
inspect(rules_sup1_conf50)
```

OUTPUT:

```
##    lhs              rhs    support  confidence lift  count
## [1] {Tiffin}     => {Coffee} 0.01058361 0.5468750 1.134577 70
## [2] {Spanish Brunch} => {Coffee} 0.01406108 0.6326531 1.312537 93
## [3] {Scone}      => {Coffee} 0.01844572 0.5422222 1.124924 122
## [4] {Toast}      => {Coffee} 0.02570305 0.7296137 1.513697 170
## [5] {Alfajores}  => {Coffee} 0.02237678 0.5522388 1.145705 148
## [6] {Juice}      => {Coffee} 0.02131842 0.5300752 1.099723 141
## [7] {Hot chocolate} => {Coffee} 0.02721500 0.5263158 1.091924 180
## [8] {Medialuna}  => {Coffee} 0.03296039 0.5751979 1.193337 218
## [9] {Cookies}    => {Coffee} 0.02978530 0.5267380 1.092800 197
## [10] {NONE}      => {Coffee} 0.04172966 0.5810526 1.205484 276
## [11] {Sandwich}  => {Coffee} 0.04233444 0.5679513 1.178303 280
## [12] {Pastry}    => {Coffee} 0.04868461 0.5590278 1.159790 322
## [13] {Cake}      => {Coffee} 0.05654672 0.5389049 1.118042 374
```

We can also create an HTML table widget using the inspectDT() function from the aruslesViz package. Rules can be interactively filtered and sorted.

C) Visualize association rules

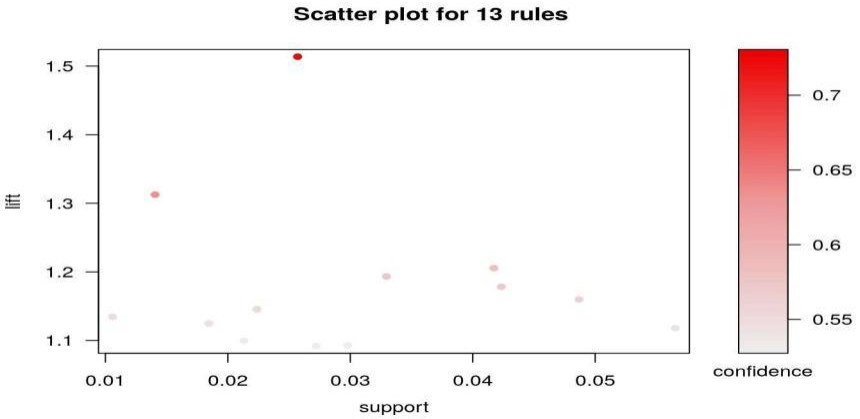
We are going to use the arulesViz package to create the visualizations. Let's begin with a simple scatter plot with different measures of interestingness on the axes (lift and support) and a third measure (confidence) represented by the color of the points.

Code

 # Scatter plot

```
plot(rules_sup1_conf50, measure=c("support", "lift"), shading="confidence")
```

OUTPUT:

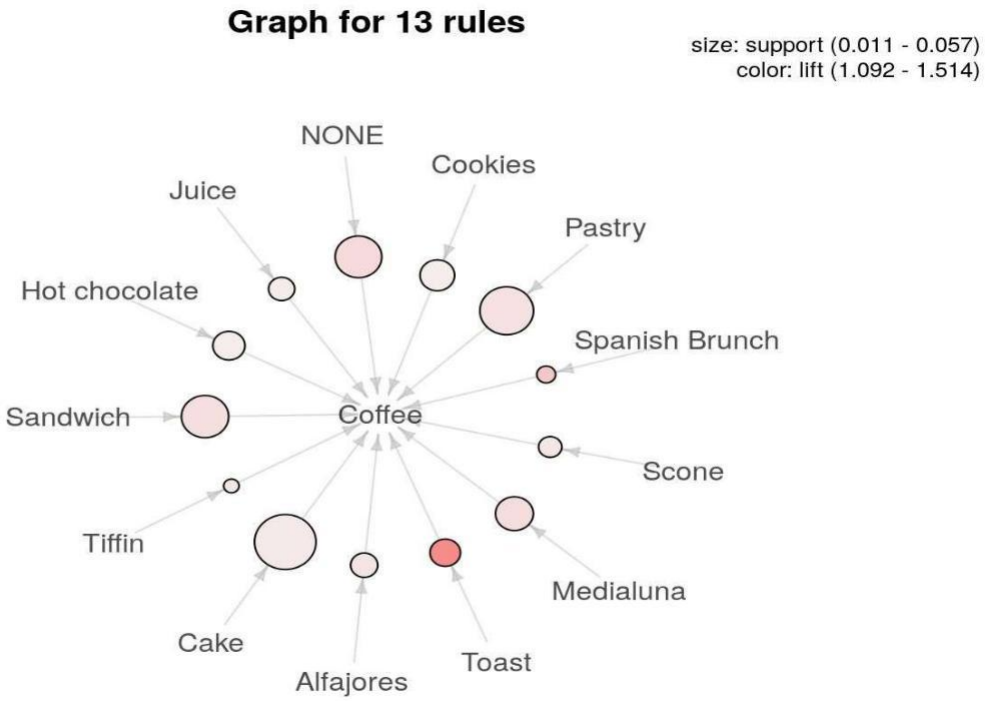


The following visualization represents the rules as a graph with items as labeled vertices, and rules represented as vertices connected to items using arrows.

Code

 # Graph (default layout)

```
plot(rules_sup1_conf50, method="graph")
```

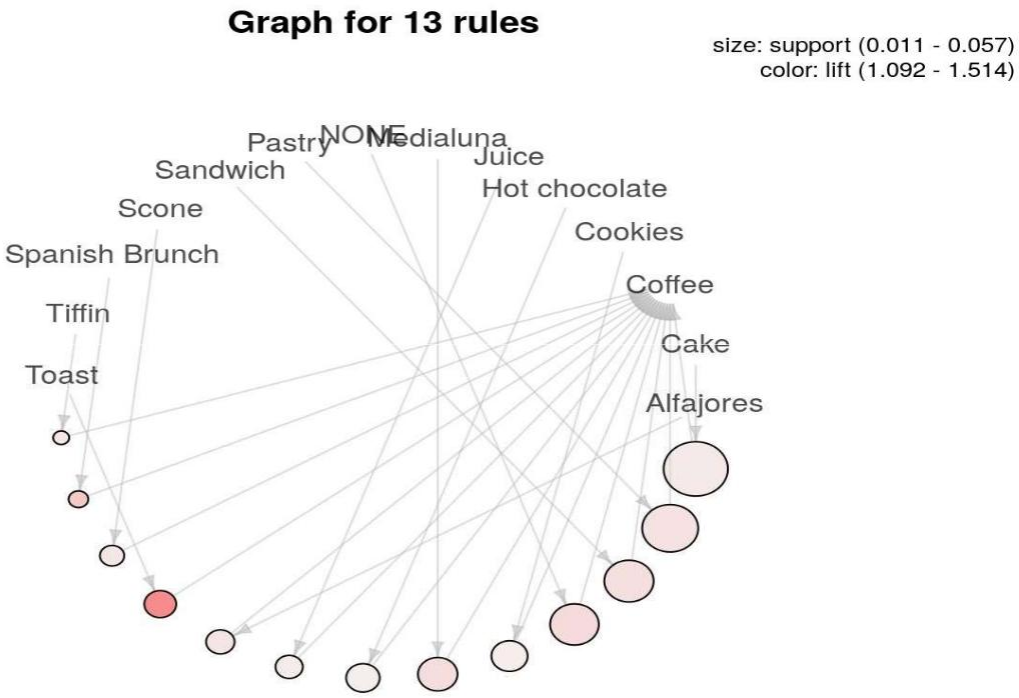


We can also change the graph layout.

Code

 # Graph (circular layout)

plot(rules_sup1_conf50, method="graph", control=list(layout=igraph::in_circle()))



What else can we do? We can represent the rules as a grouped matrix-based visualization. The support and lift measures are represented by the size and color of the ballons, respectively. In this case it's not a very useful visualization, since we only have coffe on the right-hand-side of the rules.

Code

 # Grouped matrix plot

---- **plot(rules_sup1_conf50, method="grouped")**

EVALUATION:

1. Importing Necessary Dependencies

```
import numpy as npimport pandas as pd

import plotly.express as pximport plotly.graph_objects as goimport plotly.figu
re_factory as fffrom plotly.offline import download_plotlyjs,
init_notebook_mo de, iplot

from mlxtend.preprocessing import TransactionEncoderfrom mlxtend.freque
nt_patterns import apriori, association_rulesimport networkx as nx
```

2. Loading and Reading Dataset

```
bakeryDF=pd.read_csv("../input/bakery/Bakery.csv")bakeryDF.head()
```

	TransactionNo	Items	DateTime	Daypart	DayType
0	1	Bread	2016-10-30 09:58:11	Morning	Weekend
1	2	Scandinavian	2016-10-30 10:05:34	Morning	Weekend
2	2	Scandinavian	2016-10-30 10:05:34	Morning	Weekend
3	3	Hot chocolate	2016-10-30 10:07:57	Morning	Weekend
4	3	Jam	2016-10-30 10:07:57	Morning	Weekend

```
print("Database dimension :", bakeryDF.shape)print("Database size      :", ba
keryDF.size)
```

```
Database dimension : (20507, 5)

Database size      : 102535
```

```
bakeryDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 20507 entries, 0 to 20506

Data columns (total 5 columns):

 #   Column        Non-Null Count  Dtype

---  ------        --------------  -----

 0   TransactionNo  20507 non-null  int64

 1   Items         20507 non-null  object

 2   DateTime       20507 non-null  object

 3   Daypart       20507 non-null  object

 4   DayType       20507 non-null  object

dtypes: int64(1), object(4)

memory usage: 801.2+ KB
```

```
bakeryDF['TransactionNo'].nunique()
```

```
bakeryDF.describe(include=object)
```

	Items	DateTime	Daypart	DayType

	Items	DateTime	Daypart	DayType
count	20507	20507	20507	20507
unique	94	9465	4	2
top	Coffee	2017-11-02 14:08:27	Afternoon	Weekday
freq	5471	11	11569	12807

Data Summary:

Overview

The dataset provides transaction details of all items purchased between 2016 and 2017 from the bakery online. The dataset has 20507 entries over 9000 transactions, and 4 columns.

Number of variables: 1

Numeric variables: 1

Categorical variables: 4

Number of observations: 20507

Total number of transactions: 9465

Missing cells : 0

Variables

TransactionNo : 9465 distinct values

Items has a high cardinality: 94 distinct values

DateTime has a high cardinality: 9182 distinct values

Daypart has 4 distinct values

DayType has 2 distinct values

3. Data Exploration and Visualization

3.1 Let's look into the frequent items and the best sellers

```
itemFrequency = bakeryDF['Items'].value_counts().sort_values(ascending=False)itemFrequency.head(10)
```

```
Coffee        5471
Bread         3325
Tea           1435
```

Cake 1025

Pastry 856

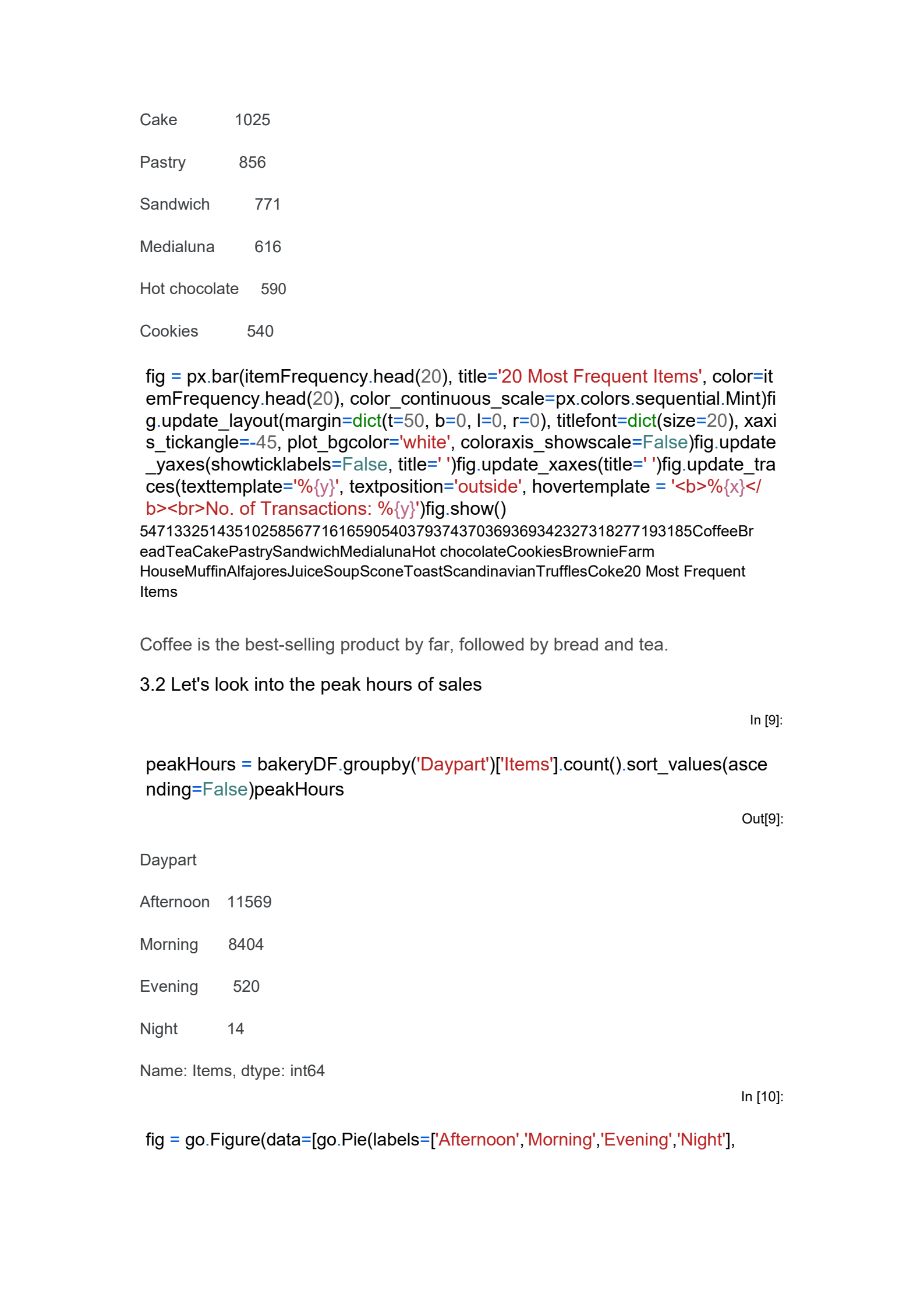
Sandwich 771

Medialuna 616

Hot chocolate 590

Cookies 540

```
fig = px.bar(itemFrequency.head(20), title='20 Most Frequent Items', color=it
emFrequency.head(20), color_continuous_scale=px.colors.sequential.Mint)fi
g.update_layout(margin=dict(t=50, b=0, l=0, r=0), titlefont=dict(size=20), xaxi
s_tickangle=-45, plot_bgcolor='white', coloraxis_showscale=False)fig.update
_yaxes(showticklabels=False, title=' ')fig.update_xaxes(title=' ')fig.update_tra
ces(texttemplate='%{y}', textposition='outside', hovertemplate = '<b>%{x}</
b><br>No. of Transactions: %{y}')fig.show()
```



Coffee is the best-selling product by far, followed by bread and tea.

3.2 Let's look into the peak hours of sales

In [9]:

```
peakHours = bakeryDF.groupby('Daypart')['Items'].count().sort_values(asce
nding=False)peakHours
```

Out[9]:

Daypart

Afternoon 11569

Morning 8404

Evening 520

Night 14

Name: Items, dtype: int64

In [10]:

```
fig = go.Figure(data=[go.Pie(labels=['Afternoon','Morning','Evening','Night'],
```

In [11]:

July 741

June 739

August 700

September 596

Name: Items, dtype: int64

```
fig = px.bar(mpm, title='Most Productive Month', color=mpm, color_continuous_scale=px.colors.sequential.Mint)fig.update_layout(margin=dict(t=50, b=0, l=0, r=0), titlefont=dict(size=20), xaxis_tickangle=0, plot_bgcolor='white', coloraxis_showscale=False)fig.update_yaxes(showticklabels=False, title=' ')fig.update_xaxes(title=' ')fig.update_traces(texttemplate='%{y}', textposition='outside', hovertemplate = '<b>%{x}</b><br>No. of Transactions: %{y}')fig.show ()
```

3220307630272748264710481041924741739700596MarchNovemberJanuaryFebruary DecemberAprilOctoberMayJulyJuneAugustSeptemberMost Productive Month

The bakery seems to be heavily occupied and makes most of its business from November to March.

EDA Summary:

Coffee is the best-selling product by far, followed by bread and tea. The bakery seems to be making most of its sales in the afternoon everyday with over 56% of the sales. Sales fall sharply after that. However the bakery makes a decent amount of sales in the morning as well. For obvious reasons, the sales are high as expected during the weekends. However the sales seem to be quite uniform rest of the days. The bakery seems to be heavily occupied and makes most of its business from November to March.

4. Association Rules Generation

4.1 Data Preparation for Association Rule Mining

Apriori algorithm requires a dataframe with all the transactions one hot encoded for all the items.

list of all the transactions

```
transactions=[]for item in bakeryDF['TransactionNo'].unique():

lst=list(set(bakeryDF[bakeryDF['TransactionNo']==item]['Items']))
```

```
    transactions.append(lst)

transactions[0:10]
```

```
[['Bread'],
```

ut[16]

```
 ['Scandinavian'],

 ['Hot chocolate', 'Jam', 'Cookies'],

 ['Muffin'],

 ['Bread', 'Coffee', 'Pastry'],

 ['Muffin', 'Pastry', 'Medialuna'],

 ['Tea', 'Coffee', 'Pastry', 'Medialuna'],

 ['Bread', 'Pastry'],

 ['Muffin', 'Bread'],

 ['Scandinavian', 'Medialuna']]
```

one hot encoding

```
te = TransactionEncoder()encodedData = te.fit(transactions).transform(trans
actions)data = pd.DataFrame(encodedData, columns=te.columns_)data.hea d()
```

	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	Toast	Truffles	Tshirt	Valentine's card	Vegan Feast	Vegan mince pie	Victoria Sponge
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	ArtTray	Bacon	Baguette	Bakewell	BarePopcorn	Basket	...	TheBART	TheNomad	Tiffin	Toast	Truffles	Tshirt	Valentine's card	VeganFeast	Veganmince pie	VictoriaSponge
			e		se	e	e	e	e	e		se	e	se	se	e	se		e	e	e
1	False	False	False	False	False	False	False	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	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

5 rows × 94 columns

4.2 Association Rules Generation

frequent items

```
frequent Items= apriori(data, use_colnames=True, min_support=0.02)frequentItems.head()
```

	support	itemsets
0	0.036344	(Alfajores)

	support	itemsets
1	0.327205	(Bread)
2	0.040042	(Brownie)
3	0.103856	(Cake)
4	0.478394	(Coffee)

```
rules = association_rules(frequentItems, metric="lift", min_threshold=1)rules.antecedents
= rules.antecedents.apply(lambda x: next(iter(x)))rules.consequents =
rules.consequents.apply(lambda x: next(iter(x)))rules.head()
```

Out[19]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	Bread	Pastry	0.327205	0.086107	0.029160	0.089119	1.034977	0.000985	1.003306
1	Pastry	Bread	0.086107	0.327205	0.029160	0.338650	1.034977	0.000985	1.017305
2	Cake	Coffee	0.103856	0.478394	0.054728	0.526958	1.101515	0.005044	1.102664
3	Coffee	Cake	0.478394	0.103856	0.054728	0.114399	1.101515	0.005044	1.011905
4	Cake	Tea	0.103856	0.142631	0.023772	0.228891	1.604781	0.008959	1.111865

4.3 Rules Visualization

In [20]:

```
network_A = list(rules["antecedents"].unique())network_B = list(rules["conse
quents"].unique())node_list = list(set(network_A + network_B))G = nx.Graph()
for i in node_list:

    G.add_node(i)for i,j in rules.iterrows():

    G.add_edges_from([(j["antecedents"], j["consequents"])])pos = nx.spring_l
ayout(G, k=0.5, dim=2, iterations=400)for n, p in pos.items():

    G.nodes[n]['pos'] = p

edge_trace = go.Scatter(x=[], y=[], line=dict(width=0.5, color='#888'),
hoverin fo='none', mode='lines')

for edge in G.edges():

    x0, y0 = G.nodes[edge[0]]['pos']

    x1, y1 = G.nodes[edge[1]]['pos']
```

```
    edge_trace['x'] += tuple([x0, x1, None])

    edge_trace['y'] += tuple([y0, y1, None])

node_trace = go.Scatter(x=[], y=[], text=[], mode='markers', hoverinfo='text',

    marker=dict(showscale=True, colorscale='Burg', reversescale=True,
color =[], size=15,

    colorbar=dict(thickness=10, title='Node Connections', xanchor='left', titlesi
de='right')))

for node in G.nodes():

    x, y = G.nodes[node]['pos']

    node_trace['x'] += tuple([x])

    node_trace['y'] += tuple([y])

for node, adjacencies in enumerate(G.adjacency()):

    node_trace['marker']['color']+=tuple([len(adjacencies[1])])

    node_info = str(adjacencies[0]) +'<br>No of Connections: {}'.format(str(len
(adjacencies[1])))

    node_trace['text']+=tuple([node_info])

fig = go.Figure(data=[edge_trace, node_trace],

    layout=go.Layout(title='Item Connections Network', titlefont=dict(size=20),

    plot_bgcolor='white', showlegend=False, margin=dict(b=0,l=0,r=0,t=50),

    xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),

    yaxis=dict(showgrid=False, zeroline=False, showticklabels=False)))

iplot(fig)
```

12345678 Node Connections Item Connections Network

Implementing market basket analysis

In [50]:

```
#Loading neccesary packagesimport numpy as npimport pandas as pdfrom mlxtend.frequent_patterns import apriorifrom mlxtend.frequent_patterns import association_rules
```

In [79]:

```
#Reading Data From Web#myretaildata = pd.read_excel('http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx')myretaildata.head()
```

Out[79]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Data Preparation

In [80]:

```
#Data Cleaningmyretaildata['Description'] = myretaildata['Description'].str.strip() #removes spaces from beginning and endmyretaildata.dropna(axis=0, subset=['InvoiceNo'], inplace=True) #removes duplicate invoicemyretaildata['InvoiceNo'] = myretaildata['InvoiceNo'].astype('str') #converting invoice number to be stringmyretaildata = myretaildata[~myretaildata['InvoiceNo'].str.contains('C')] #remove the credit transactions myretaildata.head()
```

Out[80]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [83]:

```
myretaildata['Country'].value_counts()#myretaildata.shape
```

Out[83]:

```
United Kingdom          487622
Germany                   9042
France                    8408
EIRE                      7894
Spain                     2485
Netherlands               2363
Belgium                   2031
Switzerland               1967
Portugal                  1501
Australia                 1185
Norway                    1072
Italy                      758
Channel Islands            748
Finland                    685
Cyprus                     614
Sweden                     451
Unspecified                446
Austria                    398
Denmark                    380
Poland                     330
Japan                      321
Israel                     295
Hong Kong                  284
Singapore                  222
Iceland                    182
USA                        179
Canada                     151
Greece                     145
Malta                      112
United Arab Emirates        68
European Community          60
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
```

Czech Republic 25
Bahrain 18
Saudi Arabia 9
Name: Country, dtype: int64

In [84]:

```
#Separating transactions for Germanymybasket = (myretaildata[myretaildata['Country']
=="Germany"]
      .groupby(['InvoiceNo', 'Description'])['Quantity']
      .sum().unstack().reset_index().fillna(0)
      .set_index('InvoiceNo'))
```

In [85]:

```
#viewing transaction basketmybasket.head()
```

Out[85]:

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCILS SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE SKULLS	...	YULETIDE IMAGES GIFT WRAP SET	ZINC HEART T-LIGHT HOLDER	ZINC STAR T-LIGHT HOLDER	ZINC BOX SIGN HOME	ZINC FOLKART SLEIGH BELLS	ZINC HEART LATTICE T-LIGHT HOLDER	ZINC METAL HEART DECORATION	ZINC T-LIGHT HOLDER STARS LARGE	ZINC T-LIGHT HOLDER STARS SMALL	ZINCWILLIEWINKIE CANDLE STICK
InvoiceNo																					
536527	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536840	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536861	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5369	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCILS SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE SKULLS	...	YULETIDE IMAGES GIFT WRAP SET	ZINC HEART T-LIGHT HOLDER	ZINC STAR T-LIGHT HOLDER	ZINC BOX SIGN HOME	ZINC FOLKART SLEIGH BELLS	ZINC HEART LATTICE T-LIGHT HOLDER	ZINCMETAL HEART DECORATION	ZINC T-LIGHT HOLDER STARS LARGE	ZINC T-LIGHT HOLDER STARS SMALL	ZINCWILLIEWINKIE CANDLE STICK
InvoiceNo																					
536983	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1695 columns

In [86]:

```
#converting all positive vaues to 1 and everything else to 0def my_encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
my_basket_sets = mybasket.applymap(my_encode_units)my_basket_sets.drop('POSTAGE',
inplace=True, axis=1) #Remove "postage" as an item
```

Training Model

In [87]:

```
#Generatig frequent itemsetsmy_frequent_itemsets = apriori(my_basket_sets,
min_support=0.07, use_colnames=True)
```

In [88]:

```
#generating rulesmy_rules = association_rules(my_frequent_itemsets, metric="lift",
min_threshold=1)
```

In [89]:

```
#viewing top 100 rules
my_rules.head(100)
```

Out[89]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(PLASTERS IN TIN WOODLAND ANIMALS)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.137856	0.245077	0.074398	0.539683	2.202098	0.040613	1.640006
1	(ROUND SNACK BOXES SET OF4 WOODLAND)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.245077	0.137856	0.074398	0.303571	2.202098	0.040613	1.237951
2	(ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.157549	0.245077	0.131291	0.833333	3.400298	0.092679	4.529540
3	(ROUND SNACK BOXES SET OF4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.245077	0.157549	0.131291	0.535714	3.400298	0.092679	1.814509
4	(SPACEBOY LUNCH BOX)	(ROUND SNACK BOXES SET OF4 WOODLAND)	0.102845	0.245077	0.070022	0.680851	2.778116	0.044817	2.365427
5	(ROUND SNACK BOXES SET OF4 WOODLAND)	(SPACEBOY LUNCH BOX)	0.245077	0.102845	0.070022	0.285714	2.778116	0.044817	1.256018

Making reecommendations

In [90]:

```
my_basket_sets['ROUND SNACK BOXES SET OF4 WOODLAND'].sum()
```

Out[90]:

```
112
```

In [91]:

```
my_basket_sets['SPACEBOY LUNCH BOX'].sum()
```

Out[91]:

```
47
```

In [92]:

```
#Filtering rules based on condition
my_rules[ (my_rules['lift'] >= 3) &
    (my_rules['confidence'] >= 0.3) ]
```

Out[92]:

	antecedents	consequents	anteceden t support	consequen t support	support	confidenc e	lift	leverag e	convictio n
2	(ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF4 WOODLAN D)	0.157549	0.245077	0.13129 1	0.833333	3.40029 8	0.09267 9	4.529540
3	(ROUND SNACK BOXES SET OF4 WOODLAN D)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.245077	0.157549	0.13129 1	0.535714	3.40029 8	0.09267 9	1.814509

In [ ]:

Benefits:

- Increase in sales through personalized marketing and product recommendations. Improved inventory management, reducing carrying costs and out-of-stock incidents. Enhanced customer satisfaction through a more tailored shopping experience.
- Identification of new product opportunities and potential market niches. Better fraud prevention and security measures. Conclusion: Market Basket Insight Pro 2.0 represents a significant leap forward in consumer analytics.
- By harnessing the power of cutting-edge technology and advanced analytics, businesses can gain a deeper understanding of their customers and optimize their operations to thrive in today's competitive market.
- This innovative module is poised to revolutionize how businesses analyze consumer behavior and drive success in the retail

Conclusion:

- In the quest to build a house price prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.
- We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis. Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
- Data preprocessing emerged as a pivotal aspect of thisprocess. Itinvolves cleaning, transforming, and refining the dataset to ensurethat it aligns with the requirements of machine learningalgorithms.