MARKET BASKET INSIGHTS

MEMBER: N.DHIVYA (922121106010)

NM ID: au922121106010

PROJECT TITLE: Market Basket Insights

PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION

INTRODUCTION

Market basket analysis is a data mining 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.

The adoption of market basket analysis was aided by the advent of electronic point-of-sale (POS) systems. Compared to handwritten records kept by store owners, the digital records generated by POS systems made it easier for applications to process and analyze large volumes of purchase data.

Implementation of market basket analysis requires a background in statistics and data science, as well as some algorithmic computer programming skills. For those without the needed technical skills, commercial, off-the-shelf tools exist.

Types of market basket analysis

Retailers should understand the following types of market basket analysis:

- **Predictive market basket analysis.** This type considers items purchased in sequence to determine cross-sell.
- **Differential market basket analysis.** This type considers data across different stores, as well as purchases from different customer groups during different times of the day, month or year. If a rule holds in one dimension, such as store, time period or customer group, but does not hold in the others, analysts can determine the factors responsible for the exception. These insights can lead to new product offers that drive higher sales.

Algorithms for market basket analysis

In market basket analysis, association 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.

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 analysis and is used to identify frequent items in the database, then evaluate their frequency as the datasets are expanded to larger sizes.

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:

- arulesNBMiner
- Opusminer
- RKEEL
- RSarules

Examples of market basket analysis

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.

Benefits of market basket analysis

Market basket analysis can increase sales and customer satisfaction. Using data to determine that products are often purchased together, retailers can optimize product placement, offer special deals and create new product bundles to encourage further sales of these combinations.

These improvements can generate additional sales for the retailer, while making the shopping experience more productive and valuable for customers. By using market basket analysis, customers may feel a stronger sentiment or brand loyalty toward the company.

About Dataset

Market Basket Analysis

Market basket analysis with Apriori algorithm

The retailer wants to target customers with suggestions on itemset that a customer is most likely to purchase .I was given dataset contains data of a retailer; the transaction data provides data around all the transactions that have happened over a period of time. Retailer will use result to grove in his industry and provide for customer suggestions on itemset, we be able increase customer engagement and improve customer experience and identify customer behavior. I will solve this problem with use Association Rules type of unsupervised learning technique that checks for the dependency of one data item on another data item.

Introduction

Association Rule is most used when you are planning to build association in different objects in a set. It works when you are planning to find frequent patterns in a transaction database. It can tell you what items do customers frequently buy together and it allows retailer to identify relationships between the items.

An Example of Association Rules

Assume there are 100 customers, 10 of them bought Computer Mouth, 9 bought Mat for Mouse and 8 bought both of them.

- bought Computer Mouth => bought Mat for Mouse
- support = P(Mouth & Mat) = 8/100 = 0.08
- confidence = support/P(Mat for Mouse) = 0.08/0.09 = 0.89
- lift = confidence/P(Computer Mouth) = 0.89/0.10 = 8.9
  This just simple example. In practice, a rule needs the support of several hundred transactions, before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

Strategy

- Data Import
- Data Understanding and Exploration
- Transformation of the data – so that is ready to be consumed by the association rules algorithm
- Running association rules
- Exploring the rules generated
- Filtering the generated rules
- Visualization of Rule

Dataset Description

- File name: Assignment-1_Data
- List name: retaildata
- File format: . xlsx
- Number of Row: 522065
- Number of Attributes: 7
- BillNo: 6-digit number assigned to each transaction. Nominal.
- Itemname: Product name. Nominal.
- Quantity: The quantities of each product per transaction. Numeric.
- Date: The day and time when each transaction was generated. Numeric.
- Price: Product price. Numeric.
- CustomerID: 5-digit number assigned to each customer. Nominal.
- Country: Name of the country where each customer resides. Nominal.

6	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850	United Kingdom

Libraries in R

First, we need to load required libraries. Shortly I describe all libraries.

- arules - Provides the infrastructure for representing,
  manipulating and analyzing transaction data and patterns (frequent itemsets and association rules).
- arulesViz - Extends package 'arules' with various visualization.
  techniques for association rules and item-sets. The package also includes several interactive visualizations for rule exploration.
- tidyverse - The tidyverse is an opinionated collection of R packages designed for data science.
- readxl - Read Excel Files in R.
- plyr - Tools for Splitting, Applying and Combining Data.
- ggplot2 - A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.
- knitr - Dynamic Report generation in R.
- magrittr- Provides a mechanism for chaining commands with a new forward-pipe operator, %>%. This operator will forward a value, or the result of an expression, into the next function call/expression. There is flexible support for the type of right-hand side expressions.
- dplyr - A fast, consistent tool for working with data frame like objects, both in memory and out of memory.
- tidyverse - This package is designed to make it easy to install and load multiple 'tidyverse' packages in a single step.

Data Pre-processing

Next, we need to upload Assignment-1_Data. xlsx to R to read the dataset.Now we can see our data in R.

20	536367	RECIPE BOX WITH METAL HEART	4	2010-12-01 08:34:00	7.95	13047	United Kingdom
21	536367	DOORMAT NEW ENGLAND	4	2010-12-01 08:34:00	7.95	13047	United Kingdom
22	536368	JAM MAKING SET WITH JARS	6	2010-12-01 08:34:00	4.25	13047	United Kingdom

After we will clear our data frame, will remove missing values.

To apply Association Rule mining, we need to convert dataframe into transaction data to make all items that are bought together in one invoice will be in one row. Below lines of code will combine all

products from one BillNo and Date and combine all products from that BillNo and Date as one row, with each item, separated by (,)

We don't need BillNo and Date, we will make it as Null.
Next, you have to store this transaction data into .csv

This how should look transaction data before we will go to next step.

RETROSPOT TEA SET CERAMIC 11 PC	GIRLY PINK TOOL SET	JUMBO SHOPPER VINTAGE RED PAISLEY	AIRLINE LOUNGE

At this step we already have our transaction dataset, and it shows the matrix of items which bought together. We can't see here any rules and how often it was purchase together. Now let's check how many transactions we have and what they are. We will have to have to load this transaction data into an object of the transaction class. This is done by using the R function read.transactions of the arules package. Our format of Data frame is basket.

```
35                                format = 'basket', sep=',')
```

Let's have a view our transaction object by summary(transaction)

We can see 18193 transactions (rows) and 7698 items (columns). 7698 is the product descriptions and 18193 transactions are collections of these items.

```
3 12 COLOURED PARTY BALLOONS
```

The summary gives us some useful information:

- Density tells the percentage of non-zero cells in a sparse matrix. In other words, total number of items that are purchased divided by a possible number of items in that matrix. You can calculate how many items were purchased by using density:
  18193x7698x0.002291294=337445
- Summary will show us most frequent items.
- Element (itemset/transaction) length distribution: It will gave us how many transactions are there for 1-itemset, 2-itemset and so on. The first row is telling you a number of items and the second row is telling you the number of transactions.

For example, there is only 1546 transaction for one item, 860 transactions for 2 items, and there are 419 items in one transaction which is the longest.

Let’s check item frequency plot, we will generate an itemFrequencyPlot to create an item Frequency Bar Plot to view the distribution of objects based on itemMatrix (e.g., >transactions or items in >itemsets and >rules) which is our case.

In itemFrequencyPlot(transaction,topN=20,type="absolute") first argument - our transaction object to be plotted that is tr. topN is allows us to plot top N highest frequency items. type can be as type="absolute" or type="relative". If we will chouse 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 well I made it in colure for better visualization.

Generating Rules

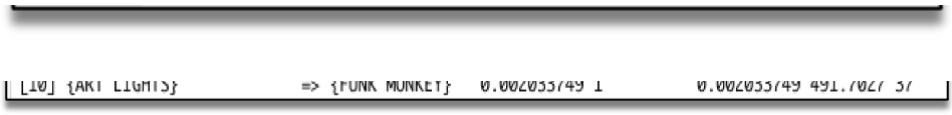
Next, we will generate rules using the Apriori algorithm. The function apriori() is from package arules. The algorithm employs level-wise search for frequent itemsets. Algorithm will generate frequent itemsets and association rules. We pass supp=0.001 and conf=0.8 to return all the rules that have a support of at least 0.1% and confidence of at least 80%. We sort the rules by decreasing confidence and will check summary of the rules.

The apriori will take (transaction) as the transaction object on which mining is to be applied. parameter will allow you to set min_sup and min_confidence. The default values for parameter are minimum support of 0.1, the minimum confidence of 0.8, maximum of 10 items (maxlen).

Summary of rules give us clear information as:

- Number of rules: 97267
- The distribution of rules by length: a length of 6 items has the most 33296 and length of 2 items has lowest number of rules 111
- The summary of quality measures: ranges of support, confidence, and lift.
- The information on data mining: total data mined, and the minimum parameters we set earlier

Now, 97267 it a lot of rules. We will identify only top 10.



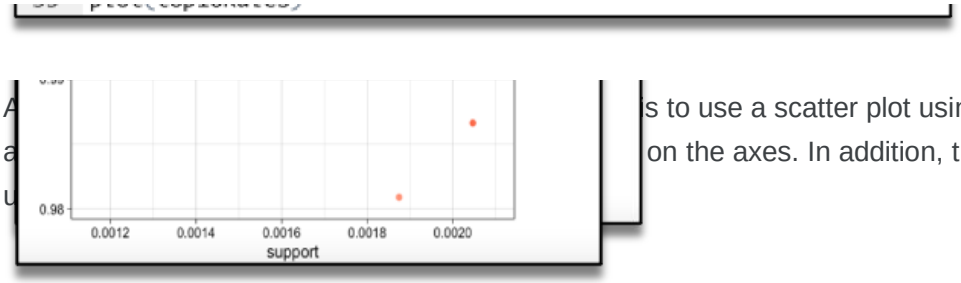
Using the above output, you can make analysis such as:

- 100% of the customers who bought 'ART LIGHTS ' also bought 'FUNK MONKEY'.
- 100% of the customers who bought 'BILLBOARD FONTS DESIGN ' also bought 'WRAP'. We can limit the size and number of rules generated. we can set parameter in Apriori. If we want stronger rules, we must to increase the value of conf. and for more extended rules give higher value to maxlen.

Visualizing Association Rules

We have thousands of rules generated based on data, we will need a couple of ways to present our findings. We will use ItemFrequencyPlot to visualize association rules.

Scatter-Plot:

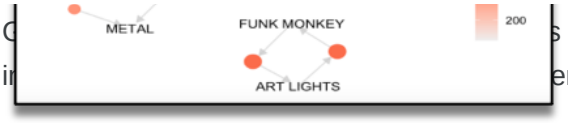


s to use a scatter plot using plot() of the
on the axes. In addition, third measure Liftis

Interactive Scatter-Plot:

We can have a look for each rule (interactively) and view all quality measures (support, confidence and lift).

Graph - Based Visualization and Group Method:



s but tend to become congested as the number of rules
er number of rules with graph-based visualizations. We

can see as well group m ethod for top 10 items

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

1. | Loading and Cleaning data

1-1. | Loading data

Out[2]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2,55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3,39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2,75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3,39	17850.0	United Kingdom

4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850.0	United Kingdom

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      522064 non-null  object
 1   Itemname    520609 non-null  object
 2   Quantity    522064 non-null  int64
 3   Date        522064 non-null  object
 4   Price       522064 non-null  object
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object
dtypes: float64(1), int64(1), object(5)
memory usage: 27.9+ MB
```

Out[4]:

```
BillNo             0
Itemname        1455
Quantity           0
Date               0
Price              0
CustomerID    134041
Country            0
dtype: int64
```

1-2. | Dropping data with negative or zero quantity

In [6]:

```
df=df.loc[df['Quantity']>0]
```

1-3. | Dropping data with zero price

In [8]:

```
df=df.loc[df['Price']>'0']
```

1-4. | Dropping Non-product data.

In [10]:

```
df=df.loc[(df['Itemname']!='POSTAGE')&(df['Itemname']!='DOTCOM
POSTAGE')&(df['Itemname']!='Adjust bad
```

```
debt')&(df['Itemname']!='Manual')]
```

1-5. | Filling null data

In [12]:

```
df=df.fillna('-')
df.isnull().sum()
```

Out[12]:

```
BillNo         0
Itemname       0
Quantity       0
Date           0
Price          0
CustomerID     0
Country        0
dtype: int64
```

1-6. | Splitting data into year and month

In [13]:

```
df['Year']=df['Date'].apply(lambda x:x.split('.')[2])
df['Year']=df['Year'].apply(lambda x:x.split(' ')[0])
df['Month']=df['Date'].apply(lambda x:x.split('.')[1])
df.head()
```

Out[13]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country	Year	Month
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2,55	17850.0	United Kingdom	2010	12
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3,39	17850.0	United Kingdom	2010	12
2	536365	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2,75	17850.0	United Kingdom	2010	12
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3,39	17850.0	United Kingdom	2010	12

4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850.0	United Kingdom	2010	12

1-7. | Creating a `Total price` column

In [14]:

```
df['Price']=df['Price'].str.replace(',','.').astype('float64')
df['Total price']=df.Quantity*df.Price
df.head()
```

Out[14]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country	Year	Month	Total price
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2.55	17850.0	United Kingdom	2010	12	15.30
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3.39	17850.0	United Kingdom	2010	12	20.34
2	536365	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2.75	17850.0	United Kingdom	2010	12	22.00
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3.39	17850.0	United Kingdom	2010	12	20.34
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3.39	17850.0	United Kingdom	2010	12	20.34

1-8. | Checking the `Total price` in each month.

In [15]:

```
df.groupby(['Year','Month'])['Total price'].sum()
```

Out[15]:

```
Year  Month
2010  12        778386.780
2011  01        648311.120
      02        490058.230
      03        659979.660
      04        507366.971
      05        721789.800
      06        710158.020
      07        642528.481
      08        701411.420
      09        981408.102
      10       1072317.070
      11       1421055.630
      12        606953.650
Name: Total price, dtype: float64
```

It is appropriate to look at 12-month increments to implement data analytics properly, so I'll drop the data for 2020 Dec.

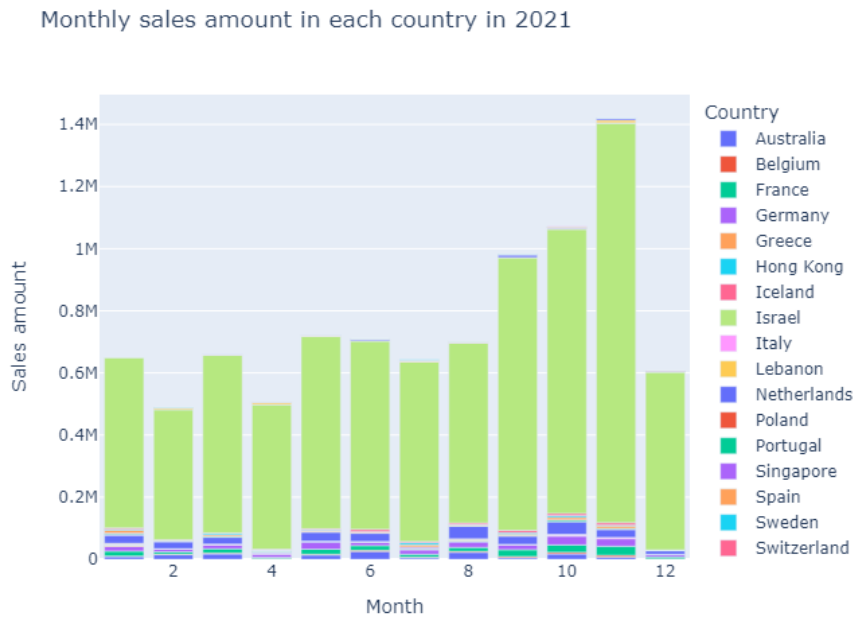
In [16]:

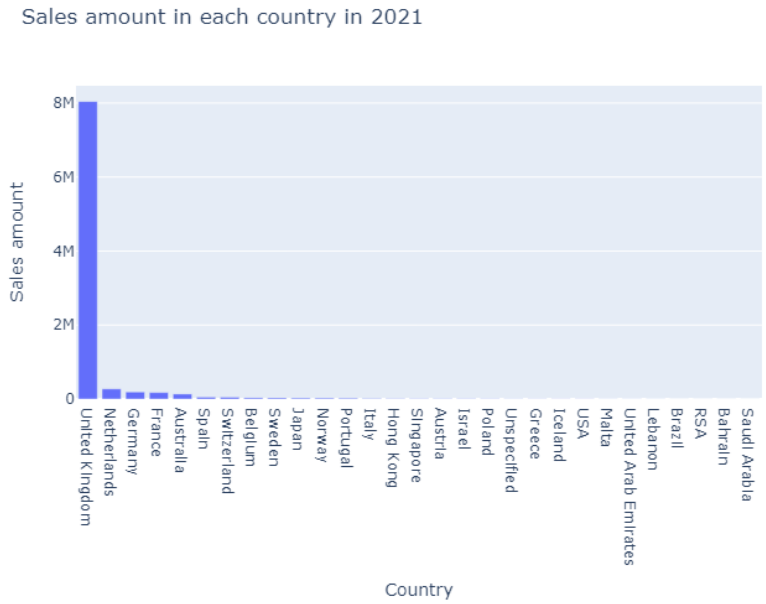
```
df=df.loc[df['Year']!='2010']
```

linkcode

2. | Exploratoty Data Analysis

2-1. | Sales amount and quantity





2-2. | **Category**

Top 10 highest sales amount items

Out[20]:

	Itemname	Price
0	REGENCY CAKESTAND 3 TIER	24653.67
1	PARTY BUNTING	9416.13
2	SET OF 3 CAKE TINS PANTRY DESIGN	7621.05
3	CREAM SWEETHEART MINI CHEST	6836.38
4	SET/4 WHITE RETRO STORAGE CUBES	6714.75
5	ENAMEL BREAD BIN CREAM	6585.93
6	WHITE HANGING HEART T-LIGHT HOLDER	6563.80
7	DOORMAT KEEP CALM AND COME IN	6385.09
8	SPOTTY BUNTING	6262.40
9	RED RETROSPOT CAKE STAND	6035.29

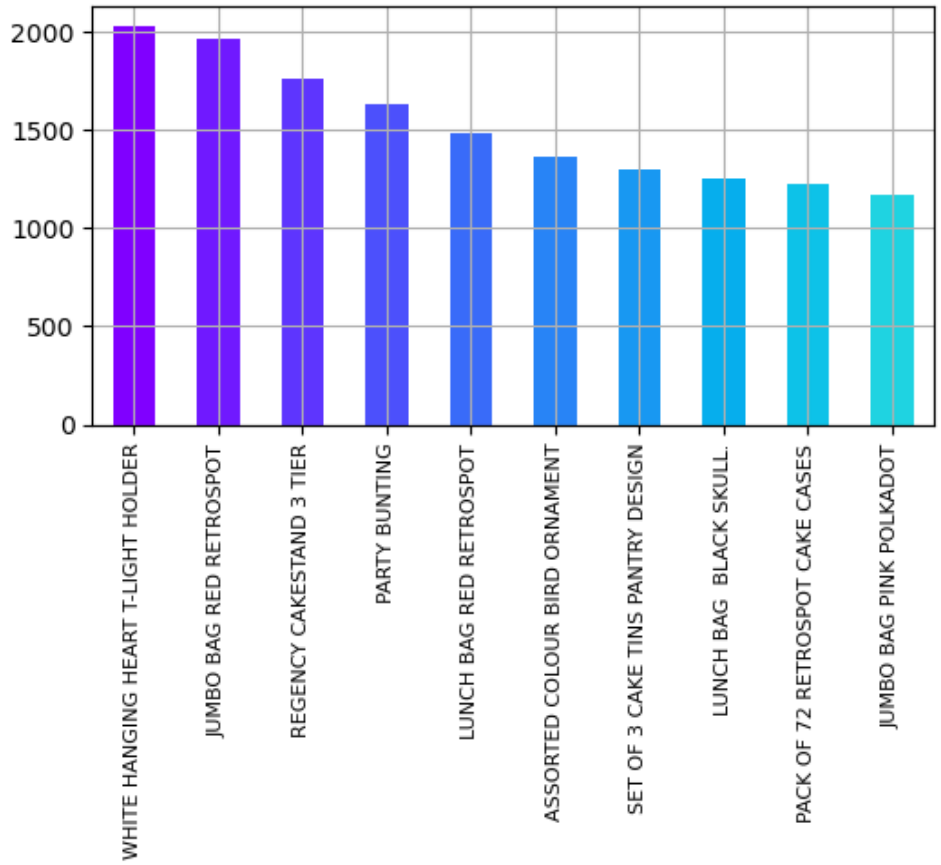
Top 10 most purchased items

Out[21]:

	Itemname	Quantity
520583	PAPER CRAFT , LITTLE BIRDIE	80995

59999	MEDIUM CERAMIC TOP STORAGE JAR	74215
405138	WORLD WAR 2 GLIDERS ASSTD DESIGNS	4800
198929	SMALL POPCORN HOLDER	4300
94245	EMPIRE DESIGN ROSETTE	3906
260928	ESSENTIAL BALM 3.5g TIN IN ENVELOPE	3186
51228	FAIRY CAKE FLANNEL ASSORTED COLOUR	3114
154834	FAIRY CAKE FLANNEL ASSORTED COLOUR	3114
416997	SMALL CHINESE STYLE SCISSOR	3000
280572	ASSORTED COLOUR BIRD ORNAMENT	2880

Top 10 most frequently purchased items



In this data, I implemented market basket analysis with apriori and specified the UK data in 2020 so that I can insight into the tendency among one year.

unfold_lessHide cell

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/market-basket-analysis/Assignment-1_Data.xlsx
/kaggle/input/market-basket-analysis/Assignment-1_Data.csv
```

1. | Loading and Cleaning data

1-1. | Loading data

Out[2]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2,55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3,39	17850.0	United Kingdom

2	536365	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2,75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3,39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850.0	United Kingdom

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   BillNo      522064 non-null  object 
 1   Itemname    520609 non-null  object 
 2   Quantity    522064 non-null  int64  
 3   Date        522064 non-null  object 
 4   Price       522064 non-null  object 
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object 
dtypes: float64(1), int64(1), object(5)
memory usage: 27.9+ MB
```

Out[4]:

```
BillNo             0
Itemname        1455
Quantity           0
Date               0
Price              0
CustomerID    134041
Country            0
dtype: int64
```

1-2. | Dropping data with negative or zero quantity

In [6]:

```
df=df.loc[df['Quantity']>0]
```

1-3. | Dropping data with zero price

In [8]:

```
df=df.loc[df['Price']>'0']
```

1-4. | Dropping Non-product data.

In [10]:

```
df=df.loc[(df['Itemname']!='POSTAGE')&(df['Itemname']!='DOTCOM
POSTAGE')&(df['Itemname']!='Adjust bad debt')&(df['Itemname']!='Manual')]
```

1-5. | Filling null data

In [12]:

```
df=df.fillna('-')
df.isnull().sum()
```

Out[12]:

```
BillNo        0
Itemname      0
Quantity      0
Date          0
Price         0
CustomerID    0
Country       0
dtype: int64
```

1-6. | Splitting data into year and month

In [13]:

```
df['Year']=df['Date'].apply(lambda x:x.split('.')[2])
df['Year']=df['Year'].apply(lambda x:x.split(' ')[0])
df['Month']=df['Date'].apply(lambda x:x.split('.')[1])
df.head()
```

Out[13]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country	Year	Month
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2,55	17850.0	United Kingdom	2010	12
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3,39	17850.0	United Kingdom	2010	12
2	536365	CREAM CUPID	8	01.12.2010 08:26	2,75	17850.0	United Kingdom	2010	12

		HEARTS COAT HANGER							
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3,39	17850.0	United Kingdom	2010	12
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850.0	United Kingdom	2010	12

1-7. | Creating a `Total price` column

In [14]:

```
df['Price']=df['Price'].str.replace(',','.').astype('float64')
df['Total price']=df.Quantity*df.Price
df.head()
```

Out[14]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country	Year	Month	Total price
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2.55	17850.0	United Kingdom	2010	12	15.30
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3.39	17850.0	United Kingdom	2010	12	20.34
2	536365	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2.75	17850.0	United Kingdom	2010	12	22.00
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3.39	17850.0	United Kingdom	2010	12	20.34
4	536365	RED WOOLLY HOTTIE	6	01.12.2010 08:26	3.39	17850.0	United Kingdom	2010	12	20.34

		WHITE HEART.								

1-8. | Checking the `Total price` in each month.

In [15]:

```
df.groupby(['Year','Month'])['Total price'].sum()
```

Out[15]:

```
Year  Month
2010  12        778386.780
2011  01        648311.120
      02        490058.230
      03        659979.660
      04        507366.971
      05        721789.800
      06        710158.020
      07        642528.481
      08        701411.420
      09        981408.102
      10       1072317.070
      11       1421055.630
      12        606953.650
Name: Total price, dtype: float64
```

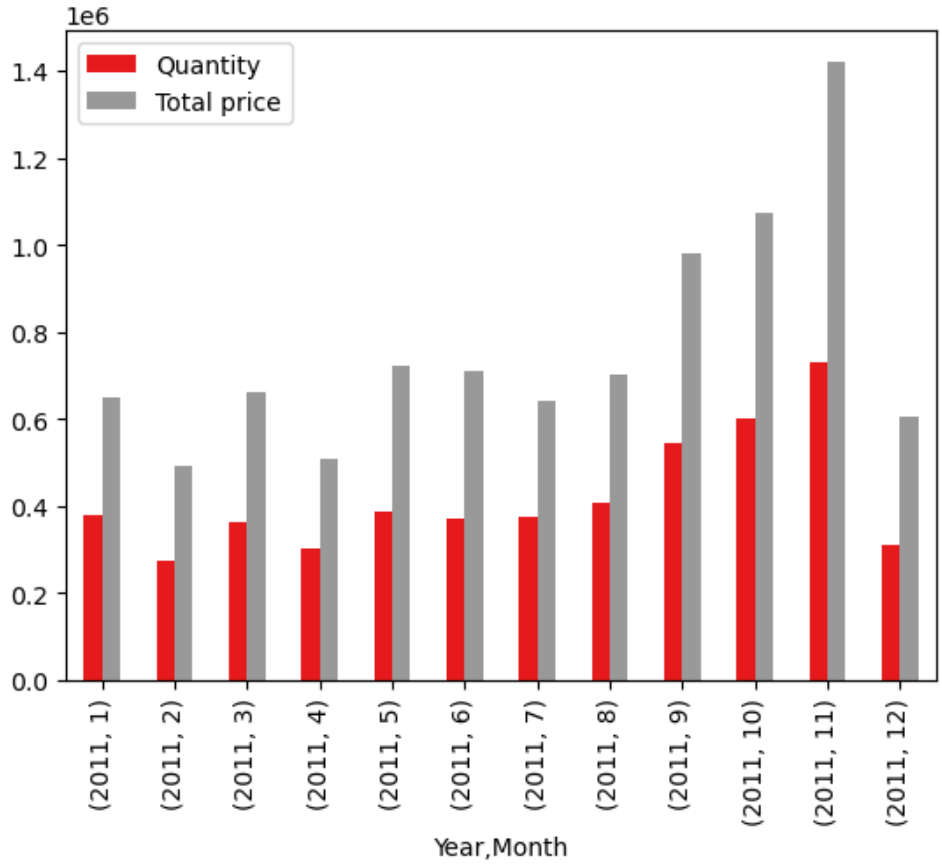
It is appropriate to look at 12-month increments to implement data analytics properly, so I'll drop the data for 2020 Dec.

In [16]:

```
df=df.loc[df['Year']!='2010']
```

2. | Exploratoty Data Analysis

2-1. | Sales amount and quantity



2468101200.2M0.4M0.6M0.8M1M1.2M1.4M

CountryAustraliaBelgiumFranceGermanyGreeceHong KongIcelandIsraelItalyLebanonNetherlandsPolandPortugalSingaporeSpainSwedenSwitzerlandUnited KingdomAustriaJapanNorwaySaudi ArabiaUnited Arab EmiratesBrazilUSAUnspecifiedBahrainMaltaRSA Monthly sales amount in each country in 2021MonthSales amount

Most of the sales amounts are occupied by the UK.

United KingdomNetherlandsGermanyFranceAustraliaSpainSwitzerlandBelgiumSwedenJapanNorwayPortugalItalyHong KongSingaporeAustriaIsraelPolandUnspecifiedGreeceIcelandUSAMaltaUnited Arab EmiratesLebanonBrazilRSABahrainSaudi Arabia02M4M6M8M

Sales amount in each country in 2021CountrySales amount

2-2. | Category

Top 10 highest sales amount items

Out[20]:

	Itemname	Price

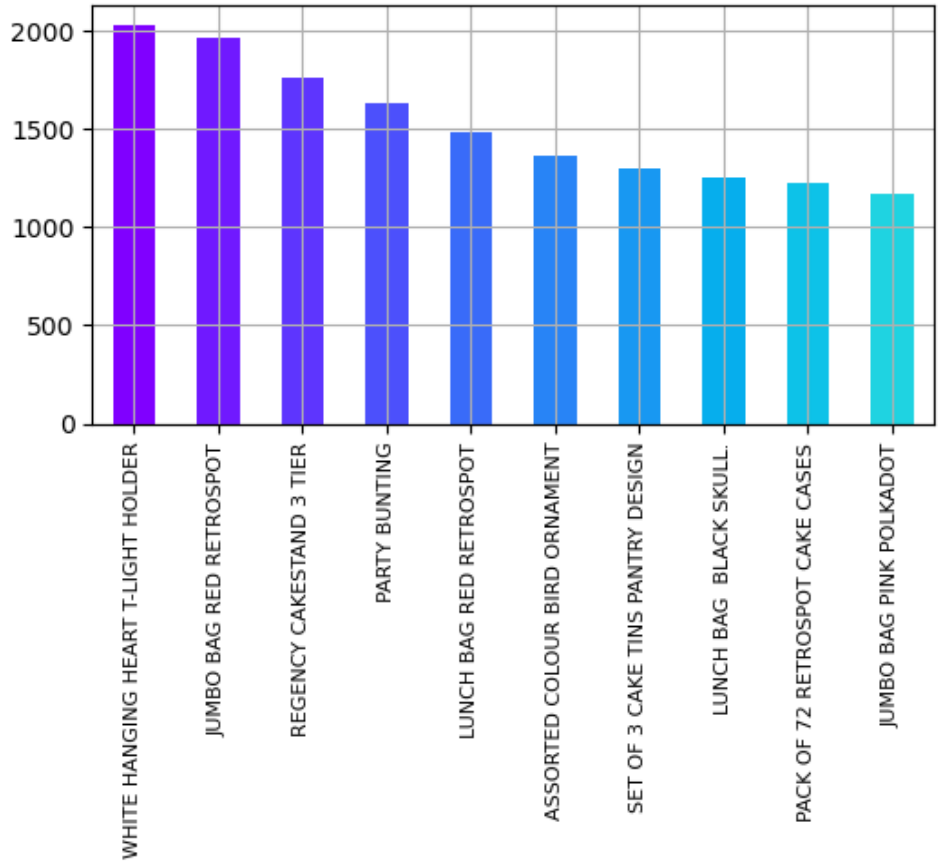
0	REGENCY CAKESTAND 3 TIER	24653.67
1	PARTY BUNTING	9416.13
2	SET OF 3 CAKE TINS PANTRY DESIGN	7621.05
3	CREAM SWEETHEART MINI CHEST	6836.38
4	SET/4 WHITE RETRO STORAGE CUBES	6714.75
5	ENAMEL BREAD BIN CREAM	6585.93
6	WHITE HANGING HEART T-LIGHT HOLDER	6563.80
7	DOORMAT KEEP CALM AND COME IN	6385.09
8	SPOTTY BUNTING	6262.40
9	RED RETROSPOT CAKE STAND	6035.29

Top 10 most purchased items

Out[21]:

	Itemname	Quantity
520583	PAPER CRAFT , LITTLE BIRDIE	80995
59999	MEDIUM CERAMIC TOP STORAGE JAR	74215
405138	WORLD WAR 2 GLIDERS ASSTD DESIGNS	4800
198929	SMALL POPCORN HOLDER	4300
94245	EMPIRE DESIGN ROSETTE	3906
260928	ESSENTIAL BALM 3.5g TIN IN ENVELOPE	3186
51228	FAIRY CAKE FLANNEL ASSORTED COLOUR	3114
154834	FAIRY CAKE FLANNEL ASSORTED COLOUR	3114
416997	SMALL CHINESE STYLE SCISSOR	3000
280572	ASSORTED COLOUR BIRD ORNAMENT	2880

Top 10 most frequently purchased items



3. | Market Basket Analysis

Since the UK is the most purchased country, let insight into the item combination purchased in the UK.

3-1. | Implementing Apriori

Out[26]:

Out[30]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.05	0.05	0.03	0.64	12.41	0.03	2.64	0.97
1	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.05	0.05	0.03	0.59	12.41	0.03	2.32	0.97
2	(GARDENERS	(GARDENERS	0.05	0.05	0.03	0.60	13.23	0.03	2.40	0.98

	KNEELING PAD KEEP CALM)	KNEELING PAD CUP OF TEA)								
3	(GARDENERS KNEELING PAD CUP OF TEA)	(GARDENERS KNEELING PAD KEEP CALM)	0.05	0.05	0.03	0.72	13.23	0.03	3.39	0.97
4	(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER)	0.04	0.05	0.03	0.82	15.50	0.03	5.25	0.98

3-2. | The top 5 of the highest **support** value of items(antecedents)

Support(item) = Transactions comprising the item / Total transactions

Out[32]:

	antecedents	consequents	support
13	frozenset({'JUMBO BAG RED RETROSPOT'})	frozenset({'JUMBO BAG PINK POLKADOT'})	0.05
12	frozenset({'JUMBO BAG PINK POLKADOT'})	frozenset({'JUMBO BAG RED RETROSPOT'})	0.05
16	frozenset({'JUMBO STORAGE BAG SUKI'})	frozenset({'JUMBO BAG RED RETROSPOT'})	0.04
17	frozenset({'JUMBO BAG RED RETROSPOT'})	frozenset({'JUMBO STORAGE BAG SUKI'})	0.04
15	frozenset({'JUMBO SHOPPER VINTAGE RED PAISLEY'})	frozenset({'JUMBO BAG RED RETROSPOT'})	0.04

In the top support value of purchase, it means that "JUMBO BAG PINK RETROSPOT" is present in 5% of all purchases.

3-3. | The top 5 of the highest **confidence** value of items

Confidence = Transactions comprising antecedent and consequent / Transactions comprising antecedent

Out[33]:

	antecedents	consequents	confidence
4	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	0.82
30	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	0.78
6	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	0.75

7	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	0.73
3	frozenset({'GARDENERS KNEELING PAD CUP OF TEA'})	frozenset({'GARDENERS KNEELING PAD KEEP CALM'})	0.72

In the top confidence value of the purchase, it means that 82% of the customers who bought "PINK REGENCY TEACUP AND SAUCER" also bought "GREEN REGENCY TEACUP AND SAUCER".

3-4. | The top 5 of the highest `lift` value of items

Lift = Confidence (antecedent -> consequent) / Support(antecedent)

In [34]:

```
rules[['antecedents','consequents','lift']].sort_values('lift',ascending=False)[:5].style.background_gradient(cmap=cm).set_precision(2)
```

Out[34]:

	antecedents	consequents	lift
4	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	15.50
5	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	15.50
31	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	14.36
30	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	14.36
6	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	13.86

In the top list value of the purchase, it means that customers are 15.5 times more likely to buy "GREEN REGENCY TEACUP AND SAUCER" if you sell "PINK REGENCY TEACUP AND SAUCER".

3-5. | The best combination of the items

Out[35]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
4	frozenset({'PINK REGENCY TEACUP AND SAUCER'})	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	0.04	0.05	0.03	0.82	15.50	0.03	5.25	0.98
30	frozenset({'PIN	frozenset({'ROS	0.04	0.05	0.03	0.78	14.36	0.03	4.24	0.97

	K REGENCY TEACUP AND SAUCER'})	ES REGENCY TEACUP AND SAUCER'})								
6	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	0.05	0.05	0.04	0.75	13.86	0.04	3.78	0.98
7	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	frozenset({'GREEN REGENCY TEACUP AND SAUCER'})	0.05	0.05	0.04	0.73	13.86	0.04	3.55	0.98
3	frozenset({'GARDENERS KNEELING PAD CUP OF TEA'})	frozenset({'GARDENERS KNEELING PAD KEEP CALM'})	0.05	0.05	0.03	0.72	13.23	0.03	3.39	0.97

As you can see above, "REGENCY TEACUP AND SAUCER" have the best combination of the same items with different colors.

4. | Conclusion

Here's what we learned from this analysis:

- **The most purchased item is `PAPER CRAFT, LITTLE BIRDIE`.**
- **The most frequently purchased item is `WHITE HANGING HEART T-LIGHT HOLDER`.**
- **The best combination items are `PINK REGENCY TEACUP AND SAUCER` and `GREEN REGENCY TEACUP AND SAUCER`.**

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
```

```
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/market-basket-analysis/Assignment-1_Data.xlsx
/kaggle/input/market-basket-analysis/Assignment-1_Data.csv
```

Market Basket Analysis Project

Overview

This notebook is part of a project focused on market basket analysis. We will begin by loading and preprocessing the dataset.

Dataset Information

The dataset is stored in the file `Assignment-1_Data.xlsx` located at `/kaggle/input/market-basket-analysis/`. It contains information related to market transactions.

Loading the Dataset

Let's start by loading the dataset into a DataFrame using pandas.

In [2]:

```
import pandas as pd

# Load the dataset
dataset_path = '/kaggle/input/market-basket-analysis/Assignment-1_Data.xlsx'
df = pd.read_excel(dataset_path)
```

Initial Exploration

We'll perform an initial exploration of the dataset to understand its structure and characteristics.

In [3]:

```
# Display basic information about the dataset
print("Number of rows and columns:", df.shape)
print("\nData Types and Missing Values:")
print(df.info())
print("\nFirst few rows of the dataset:")
print(df.head())
```

```
Number of rows and columns: (522064, 7)

Data Types and Missing Values:
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      522064 non-null  object
 1   Itemname    520609 non-null  object
 2   Quantity    522064 non-null  int64
 3   Date        522064 non-null  datetime64[ns]
 4   Price       522064 non-null  float64
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 27.9+ MB
None

First few rows of the dataset:
   BillNo                          Itemname  Quantity
Date  \
0  536365   WHITE HANGING HEART T-LIGHT HOLDER         6 2010-12-01
08:26:00
1  536365                  WHITE METAL LANTERN         6 2010-12-01
08:26:00
2  536365       CREAM CUPID HEARTS COAT HANGER         8 2010-12-01
08:26:00
3  536365  KNITTED UNION FLAG HOT WATER BOTTLE         6 2010-12-01
08:26:00
4  536365       RED WOOLLY HOTTIE WHITE HEART.         6 2010-12-01
08:26:00

   Price  CustomerID         Country
0   2.55     17850.0  United Kingdom
1   3.39     17850.0  United Kingdom
2   2.75     17850.0  United Kingdom
3   3.39     17850.0  United Kingdom
4   3.39     17850.0  United Kingdom
```

Preprocessing

We'll preprocess the data to ensure it's ready for analysis.

In [4]:

```
#Check Missing Values
print("Missing Values:")
```

```
print(df.isnull().sum())

#Drop Rows with Missing Values
df.dropna(inplace=True)
```

```
Missing Values:
BillNo             0
Itemname        1455
Quantity           0
Date               0
Price              0
CustomerID    134041
Country            0
dtype: int64
```

In [5]:

```
# Convert dataframe into transaction data
transaction_data = df.groupby(['BillNo', 'Date'])['Itemname'].apply(lambda
x: ', '.join(x)).reset_index()

#Drop Unnecessary Columns
columns_to_drop = ['BillNo', 'Date']
transaction_data.drop(columns=columns_to_drop, inplace=True)

# Save the transaction data to a CSV file
transaction_data_path = '/kaggle/working/transaction_data.csv'
transaction_data.to_csv(transaction_data_path, index=False)
```

In [6]:

```
# Display the first few rows of the transaction data
print("\nTransaction Data for Association Rule Mining:")
print(transaction_data.head())
transaction_data.shape
```

```
Transaction Data for Association Rule Mining:
                                            Itemname
0  WHITE HANGING HEART T-LIGHT HOLDER, WHITE META...
1  HAND WARMER UNION JACK, HAND WARMER RED POLKA DOT
2  ASSORTED COLOUR BIRD ORNAMENT, POPPY'S PLAYHOU...
3  JAM MAKING SET WITH JARS, RED COAT RACK PARIS ...
4                          BATH BUILDING BLOCK WORD
```

Out[6]:

```
(18192, 1)
```

Phase 4 starts from here

Formatting the transaction data in a suitable format for analysis

Developing the preprocessed data into analysis. Split the 'Itemname' column in `transaction_data` into individual items using `str.split(', ', expand=True)`.Concatenate the original DataFrame (`transaction_data`) with the items DataFrame (`items_df`) using `pd.concat`.Drop the original 'Itemname' column since individual items are now in separate columns.Display the resulting DataFrame.

In [7]:

```
# Split the 'Itemname' column into individual items
items_df = transaction_data['Itemname'].str.split(', ', expand=True)

# Concatenate the original DataFrame with the new items DataFrame
transaction_data = pd.concat([transaction_data, items_df], axis=1)

# Drop the original 'Itemname' column
transaction_data = transaction_data.drop('Itemname', axis=1)

# Display the resulting DataFrame
print(transaction_data.head())
```

```
                                0                               1    \
0  WHITE HANGING HEART T-LIGHT HOLDER          WHITE METAL LANTERN
1              HAND WARMER UNION JACK    HAND WARMER RED POLKA DOT
2       ASSORTED COLOUR BIRD ORNAMENT    POPPY'S PLAYHOUSE BEDROOM
3            JAM MAKING SET WITH JARS  RED COAT RACK PARIS FASHION
4            BATH BUILDING BLOCK WORD                          None

                             2                                    3    \
0  CREAM CUPID HEARTS COAT HANGER  KNITTED UNION FLAG HOT WATER BOTTLE
1                            None                                 None
2       POPPY'S PLAYHOUSE KITCHEN    FELTCRAFT PRINCESS CHARLOTTE DOLL
3  YELLOW COAT RACK PARIS FASHION         BLUE COAT RACK PARIS FASHION
4                            None                                 None

                             4                                    5    \
0  RED WOOLLY HOTTIE WHITE HEART.        SET 7 BABUSHKA NESTING BOXES
1                            None                                 None
2          IVORY KNITTED MUG COSY  BOX OF 6 ASSORTED COLOUR TEASPOONS
3                            None                                 None
```

```
4                                None                                  None

                                   6                                   7    \
0  GLASS STAR FROSTED T-LIGHT HOLDER                                None
1                                None                                None
2       BOX OF VINTAGE JIGSAW BLOCKS  BOX OF VINTAGE ALPHABET BLOCKS
3                                None                                None
4                                None                                None

                         8                         9    ...   534   535
536  \
0                     None                      None  ...  None  None
None
1                     None                      None  ...  None  None
None
2  HOME BUILDING BLOCK WORD  LOVE BUILDING BLOCK WORD  ...  None  None
None
3                     None                      None  ...  None  None
None
4                     None                      None  ...  None  None
None

    537   538   539   540   541   542   543
0  None  None  None  None  None  None  None
1  None  None  None  None  None  None  None
2  None  None  None  None  None  None  None
3  None  None  None  None  None  None  None
4  None  None  None  None  None  None  None

[5 rows x 544 columns]
```

Association Rules - Data Mining

Converting Items to Boolean Columns

To prepare the data for association rule mining, we convert the items in the `transaction_data` DataFrame into boolean columns using one-hot encoding. This is achieved through the `pd.get_dummies` function, which creates a new DataFrame (`df_encoded`) with boolean columns representing the presence or absence of each item.

In [8]:

```
# Convert items to boolean columns
df_encoded = pd.get_dummies(transaction_data, prefix='',
```

```
prefix_sep='').groupby(level=0, axis=1).max()

# Save the transaction data to a CSV file
df_encoded.to_csv('transaction_data_encoded.csv', index=False)
```

Association Rule Mining

We apply the Apriori algorithm to perform association rule mining on the encoded transaction data. The `min_support` parameter is set to 0.007 to filter out infrequent itemsets. The resulting frequent itemsets are then used to generate association rules based on a minimum confidence threshold of 0.5.Finally, we print the generated association rules.

In [9]:

```
# Load transaction data into a DataFrame
df_encoded = pd.read_csv('transaction_data_encoded.csv')

from mlxtend.frequent_patterns import apriori, association_rules

# Association Rule Mining
frequent_itemsets = apriori(df_encoded, min_support=0.007,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.5)

# Display information of the rules
print("Association Rules:")
print(rules.head())
```

```
Association Rules:
                        antecedents                        consequents  \
0            (CHOCOLATE BOX RIBBONS)          (6 RIBBONS RUSTIC CHARM)
1  (60 CAKE CASES DOLLY GIRL DESIGN)  (PACK OF 72 RETROSPOT CAKE CASES)
2      (60 TEATIME FAIRY CAKE CASES)  (PACK OF 72 RETROSPOT CAKE CASES)
3   (ALARM CLOCK BAKELIKE CHOCOLATE)       (ALARM CLOCK BAKELIKE GREEN)
4   (ALARM CLOCK BAKELIKE CHOCOLATE)        (ALARM CLOCK BAKELIKE PINK)

   antecedent support  consequent support   support  confidence       lift
\
0            0.012368            0.039193  0.007036    0.568889  14.515044
1            0.018525            0.054529  0.010059    0.543027   9.958409
2            0.034631            0.054529  0.017315    0.500000   9.169355
3            0.017150            0.042931  0.011379    0.663462  15.454151
4            0.017150            0.032652  0.009125    0.532051  16.294742
```

```
   leverage  conviction  zhangs_metric
0  0.006551    2.228676       0.942766
1  0.009049    2.068984       0.916561
2  0.015427    1.890941       0.922902
3  0.010642    2.843862       0.951613
4  0.008565    2.067210       0.955009
```

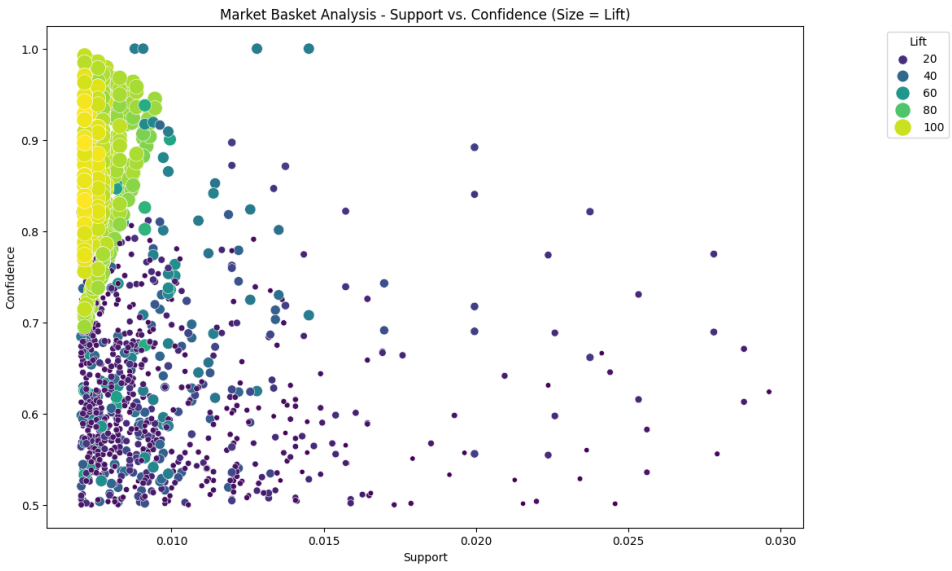
Visualization

Visualizing Market Basket Analysis Results

We use matplotlib and seaborn libraries to create a scatterplot visualizing the results of the market basket analysis. The plot depicts the relationship between support, confidence, and lift for the generated association rules.

In [10]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot scatterplot for Support vs. Confidence
plt.figure(figsize=(12, 8))
sns.scatterplot(x="support", y="confidence", size="lift", data=rules,
hue="lift", palette="viridis", sizes=(20, 200))
plt.title('Market Basket Analysis - Support vs. Confidence (Size = Lift)')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.legend(title='Lift', loc='upper right', bbox_to_anchor=(1.2, 1))
plt.show()
```

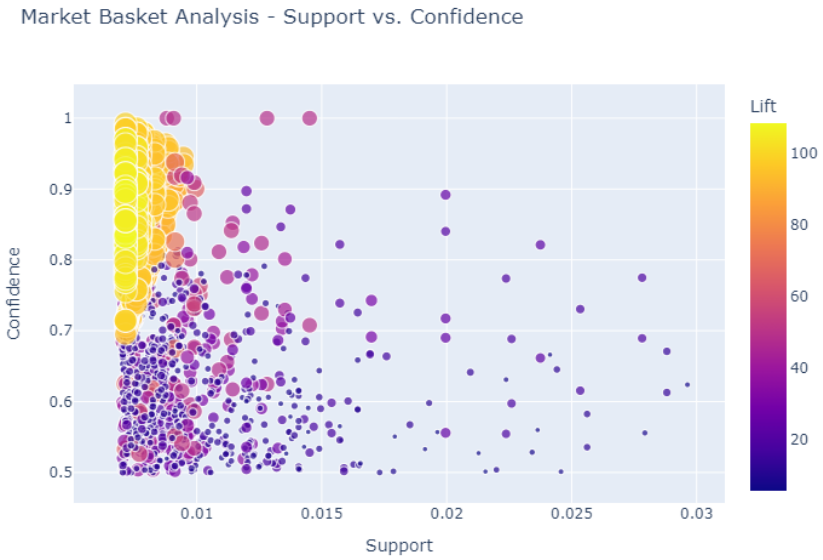


Interactive Market Basket Analysis Visualization

We leverage the Plotly Express library to create an interactive scatter plot visualizing the results of the market basket analysis. This plot provides an interactive exploration of the relationship between support, confidence, and lift for the generated association rules.

In [11]:

```
import plotly.express as px

# Convert frozensets to lists for serialization
rules['antecedents'] = rules['antecedents'].apply(list)
rules['consequents'] = rules['consequents'].apply(list)

# Create an interactive scatter plot using plotly express
fig = px.scatter(rules, x="support", y="confidence", size="lift",
                 color="lift", hover_name="consequents",
                 title='Market Basket Analysis - Support vs. Confidence',
                 labels={'support': 'Support', 'confidence':
'Confidence'})

# Customize the layout
fig.update_layout(
    xaxis_title='Support',
    yaxis_title='Confidence',
    coloraxis_colorbar_title='Lift',
    showlegend=True
)

# Show the interactive plot
fig.show()
```



Interactive Network Visualization for Association Rules

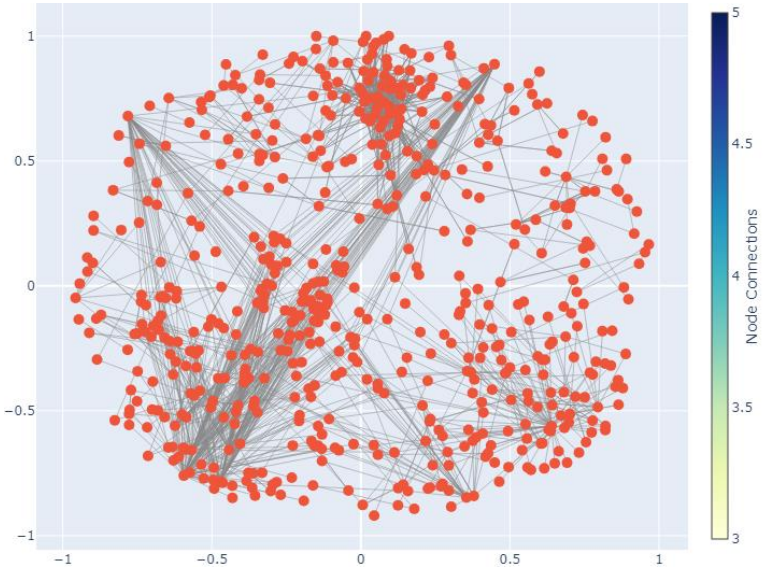
We utilize the NetworkX and Plotly libraries to create an interactive network graph visualizing the association rules. This graph represents relationships between antecedent and consequent items, showcasing support as edge weights.

In [12]:

```
import networkx as nx
import matplotlib.pyplot as plt
import plotly.graph_objects as go

# Create a directed graph
G = nx.DiGraph()

# Add nodes and edges from association rules
for idx, row in rules.iterrows():
    G.add_node(tuple(row['antecedents']), color='skyblue')
    G.add_node(tuple(row['consequents']), color='orange')
    G.add_edge(tuple(row['antecedents']), tuple(row['consequents']),
weight=row['support'])

# Set node positions using a spring layout
pos = nx.spring_layout(G)

# Create an interactive plot using plotly
```

```
edge_x = []
edge_y = []
for edge in G.edges(data=True):
    x0, y0 = pos[edge[0]]
    x1, y1 = pos[edge[1]]
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
for node in G.nodes():
    x, y = pos[node]
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    marker=dict(
        showscale=True,
        colorscale='YlGnBu',
        size=10,
        colorbar=dict(
            thickness=15,
            title='Node Connections',
            xanchor='left',
            titleside='right'
        )
    )
)

# Customize the layout
```

```
layout = go.Layout(
    showlegend=False,
    hovermode='closest',
    margin=dict(b=0, l=0, r=0, t=0),
)

# Create the figure
fig = go.Figure(data=[edge_trace, node_trace], layout=layout)

# Show the interactive graph
fig.show()
```



Interactive Sunburst Chart for Association Rules

We use Plotly Express to create an interactive sunburst chart visualizing association rules. This chart represents the relationships between antecedent and consequent items, showcasing lift as well as support through color intensity.

In [13]:

linkcode

```
import plotly.express as px

# Combine antecedents and consequents into a single column for each rule
rules['rule'] = rules['antecedents'].astype(str) + ' -> ' +
```

```
rules['consequents'].astype(str)

# Create a sunburst chart
fig = px.sunburst(rules, path=['rule'], values='lift',
                  title='Market Basket Analysis - Sunburst Chart',
                  color='support', color_continuous_scale='YlGnBu')

# Customize the layout
fig.update_layout(
    margin=dict(l=0, r=0, b=0, t=40),
)

# Show the interactive plot
fig.show()
```