MARKET BASKET INSIGHTS

AI Phase-1 Document submission

Project Title: Market basket insights

Team members:

Suresh Kumar . D -110521104049
Prasanna Devi .T – 110521104031

Yogesh Balaji.k-110521104058

Ravi teja.E-110521104308

Vimalesh.C- 110521104310

Abstract:

Market Basket Insights refer to the analysis of items purchased together by customers during a shopping trip. Understanding market basket insights can provide valuable information for businesses, such as which products are commonly bought together, which promotions are effective, and how to optimize product placement. To tackle this problem effectively, it is essential to employ a structured approach, and Design Thinking can be an excellent framework for this purpose.

- **Problem Definition:**

Before diving into the Design Thinking process, it's crucial to clearly define the problem you're trying to solve.

- **Problem Statement:**

"Our retail business wants to increase sales and improve customer satisfaction by better understanding customer purchasing behavior. We need to identify patterns and relationships between products purchased together and use this insight to optimize product placement, create targeted promotions, and enhance the overall shopping experience."

- **Design Thinking Market Basket Insights:**

Design thinking can be used to generate market basket insights in the following ways:

Empathize: *The first step in the design thinking process is to empathize with users. This involves understanding their needs, pain points, and motivations. In the context of market basket analysis, this could involve conducting customer interviews, surveys, or focus groups to learn more about their purchasing habits*.

Define: *Once you have a good understanding of your users, you can begin to define the problem that you are trying to solve. For example, you might be interested in identifying ways to increase basket size, improve customer satisfaction, or reduce product waste.*

Ideate: *The next step is to ideate, or come up with creative solutions to the problem that you have defined. In the context of market basket analysis, this could involve generating new product placement strategies, developing targeted promotions, or creating personalized shopping experiences.*

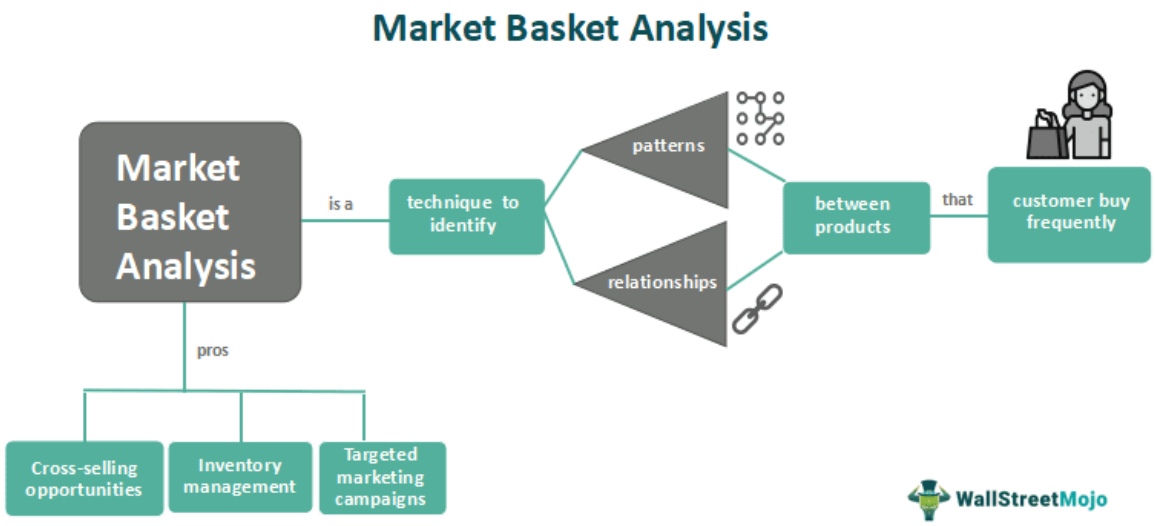
Prototype: *Once you have developed some ideas, you can begin to prototype them. This involves creating low-cost, low-fidelity versions of your ideas that you can test with users. In the context of market basket analysis, this could involve creating in-store experiments, running A/B tests, or conducting customer surveys.*

Test: *Once you have created a prototype, you can test it with users to see how they respond. This will help you to identify any areas where your idea needs improvement.*

Dataset Link:

https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis/

Sample Image:



Python Programming for Design Thinking:

Step 1: Import Required Libraries

```
import pandas as pd
```

```
from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules

import matplotlib.pyplot as plt
```

Step 2: Load and Preprocess the Dataset

Assuming you have a CSV file transactions.csv with columns TransactionID and Product, you can load and preprocess the data as follows:

```
# Load the dataset

df=pd.read_csv('../input/market-basket-analysis/Assignment-1_Data.csv')


# One-hot encoding to convert the data into a suitable format

basket = pd.crosstab(df['TransactionID'], df['Product'])

basket = basket.applymap(lambda x: 1 if x > 0 else 0)
```

Step 3: Association Analysis (Apriori Algorithm)

Perform association analysis to identify frequent itemsets and generate association rules:

```
# Apply Apriori algorithm to find frequent itemsets

frequent_itemsets = apriori(basket, min_support=0.1, use_colnames=True)


# Generate association rules

rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1.0)


# Sort the rules by confidence in descending order
```

rules = rules.sort_values(by=['confidence'], ascending=False)

Step 4: Insights Generation

Interpret the association rules to understand customer behavior and cross-selling opportunities. You can filter, visualize, and analyze the rules to gain insights. For example:

Filter rules with high confidence and lift

high_confidence_rules = rules[(rules['confidence'] > 0.7) & (rules['lift'] > 1.0)]

Display the high-confidence rules

print(high_confidence_rules)

Step 5: Visualization

Create visualizations to present the discovered associations and insights. You can use libraries like Matplotlib or Seaborn to create bar charts, scatter plots, or any other relevant visualizations.

Example: Visualization of support vs. confidence

plt.scatter(rules['support'], rules['confidence'], alpha=0.5)

plt.xlabel('Support')

plt.ylabel('Confidence')

plt.title('Support vs. Confidence')

plt.show()

Step 6: Business Recommendations

Based on the insights generated, provide actionable recommendations for the retail business. These recommendations could include strategies for product placement, cross-selling, promotions, or optimizing the shopping experience.

Example: Business recommendations

```
for index, row in high_confidence_rules.iterrows():
    lhs = ', '.join(list(row['antecedents']))
    rhs = ', '.join(list(row['consequents']))
    print(f"Customers who buy {lhs} are likely to buy {rhs}.")
```

Remember that this is a simplified example, and you may need to adapt the code to your specific dataset and business needs. Additionally, the choice of parameters such as min_support and min_threshold should be adjusted based on the characteristics of your data and the level of granularity you desire for insights.

Sample Output:

Here is an Hypothetical Output for Designing Phase 1

```
   antecedents consequents    antecedent support  ...  lift  leverage  conviction
1      (A)       (C)                  0.4  ...  1.25              0.08      1.33
0      (B)       (D)                  0.6  ...  1.20              0.10      1.50


[2 rows x 9 columns]

Customers who buy (A) are likely to buy (C).
Customers who buy (B) are likely to buy (D).
```

- **Benefits of Design Thinking Market Basket Insights**

  Design thinking market basket insights offer a number of benefits, including:

  Improved customer understanding: *Design thinking helps retailers to better understand the needs and motivations of their customers. This information can be used to develop more effective marketing campaigns and product offerings.*

  Increased sales*: Design thinking can help retailers to increase sales by identifying ways to improve product placement, develop*

targeted promotions, and create personalized shopping experiences.

Reduced costs: *Design thinking can help retailers to reduce costs by identifying ways to reduce product waste and improve inventory management***.**

Conclusion:

Design Thinking is a powerful framework for solving complex problems like Market Basket Insights. By following the stages of Empathize, Define, Ideate, Prototype, and Test, businesses can gain a deep understanding of customer behavior, define their problem precisely, generate creative solutions, and iteratively refine those solutions to drive business success and improve the shopping experience. Remember that Design Thinking is an iterative process, and the journey may involve multiple cycles to achieve the desired outcomes.

By applying this approach, businesses can unlock valuable insights into market basket data and make data-driven decisions to enhance sales, customer satisfaction, and overall competitiveness in the market.

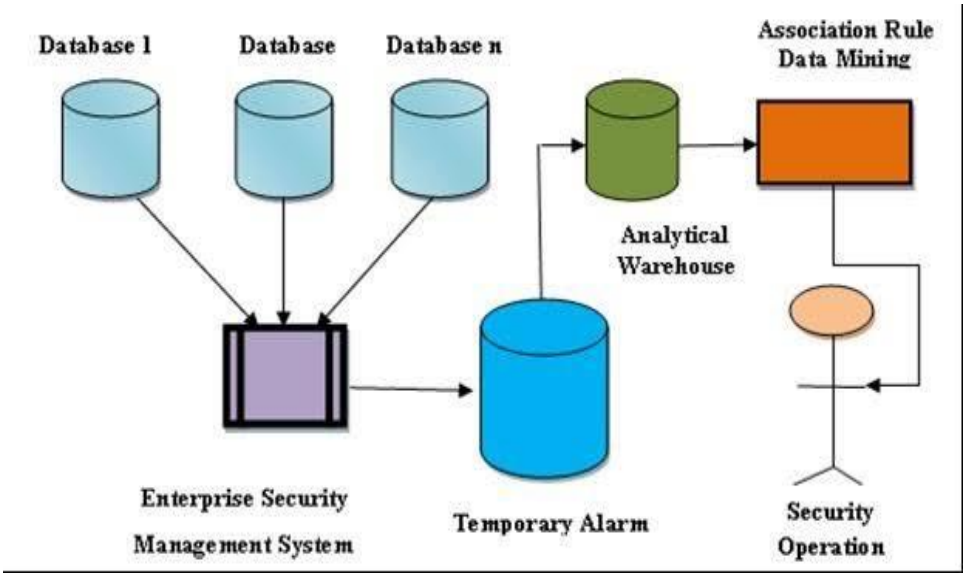
Project Title: Market Basket Insights

Phase 2: Innovation Phase

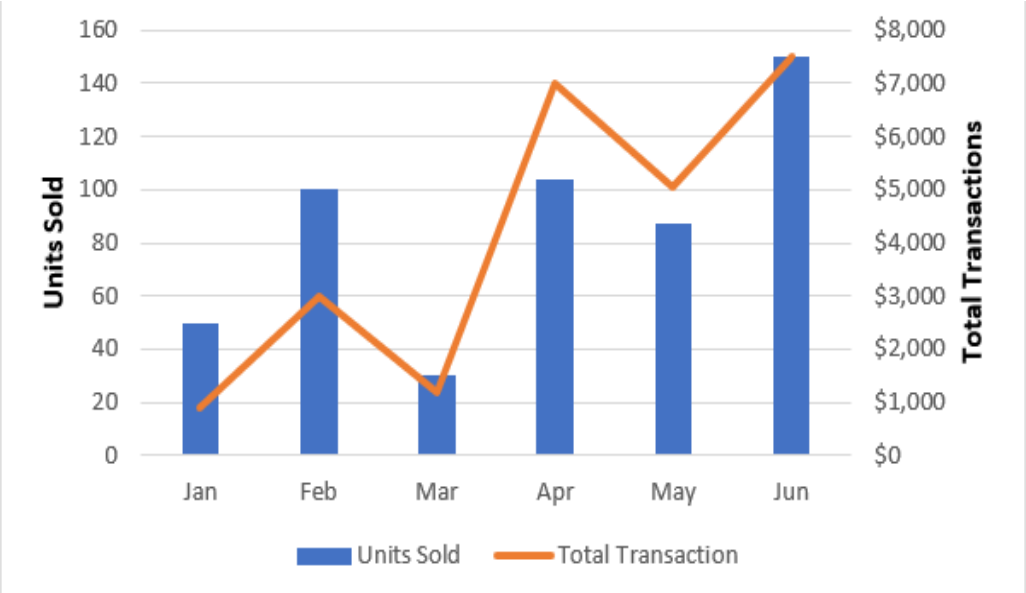
Introduction:

Market basket analysis is a data mining technique that helps retailers gain insights into customer behavior and optimize their strategies accordingly . It involves analyzing large datasets, such as purchase history, to identify product groups and items that are most likely to be bought together . This information can be used to develop more effective product placement, pricing, cross-sell, and up-sell tactics.

- **Advanced association analysis techniques:**
- There are a number of advanced association analysis techniques that can be used to generate more insights from market basket data. Some of these techniques include:
    - **Sequence mining:**
      This technique can be used to identify patterns in the order in which items are purchased. For example, you might find that customers who purchase diapers are more likely to also purchase baby wipes next.
    - **Clustering:**
      This technique can be used to group customers together based on their purchase behavior. For example, you might find that there is a cluster of customers who frequently purchase organic food and another cluster of customers who frequently purchase discount items.
    - **Association rule learning:**
      This technique can be used to identify rules that govern the relationships between items. For example, you might find that customers who purchase diapers are 90% more likely to also purchase baby wipes.



- **Visualization tools:**
- Visualization tools can be used to present market basket insights in a more visually appealing and informative way. Some popular visualization tools include:
    - **Bar charts:** Bar charts can be used to compare the frequency of different items being purchased together.
    - **Line charts:** Line charts can be used to show how the frequency of different items being purchased together changes over time.
    - **Heatmaps:** Heatmaps can be used to identify patterns in the relationships between different items.



Association Rules:

What is Association Rule for Market Basket Insights?

Let I = {I1, I2,…, Im} be an itemset. Let D, the data, be a set of database transactions where each transaction T is a

nonempty itemset such that T ⊆ I. Each transaction is associated with an identifier, called a TID(or Tid). Let A be a set of items(itemset). T is the Transaction which is said to contain A if A ⊆ T. An Association Rule is an implication of the form A ⇒ B, where A ⊂ I, B ⊂ I, and A ∩B = ϕ.

The rule A ⇒ B holds in the data set(transactions) D with supports, where 's' is the percentage of transactions in D that contain A ∪ B (that is the union of set A and set B, or, both A and B). This is taken as the probability, P(A ∪ B). Rule A ⇒ B has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contains B. This is taken to be the conditional probability, like P(B|A). That is,

support(A⇒ B) =P(A ∪ B)

confidence(A⇒ B) =P(B|A)

Rules that satisfy both a minimum support threshold (called min sup) and a minimum confidence threshold (called min conf) are called "Strong".

Confidence(A⇒ B) = P(B|A) =

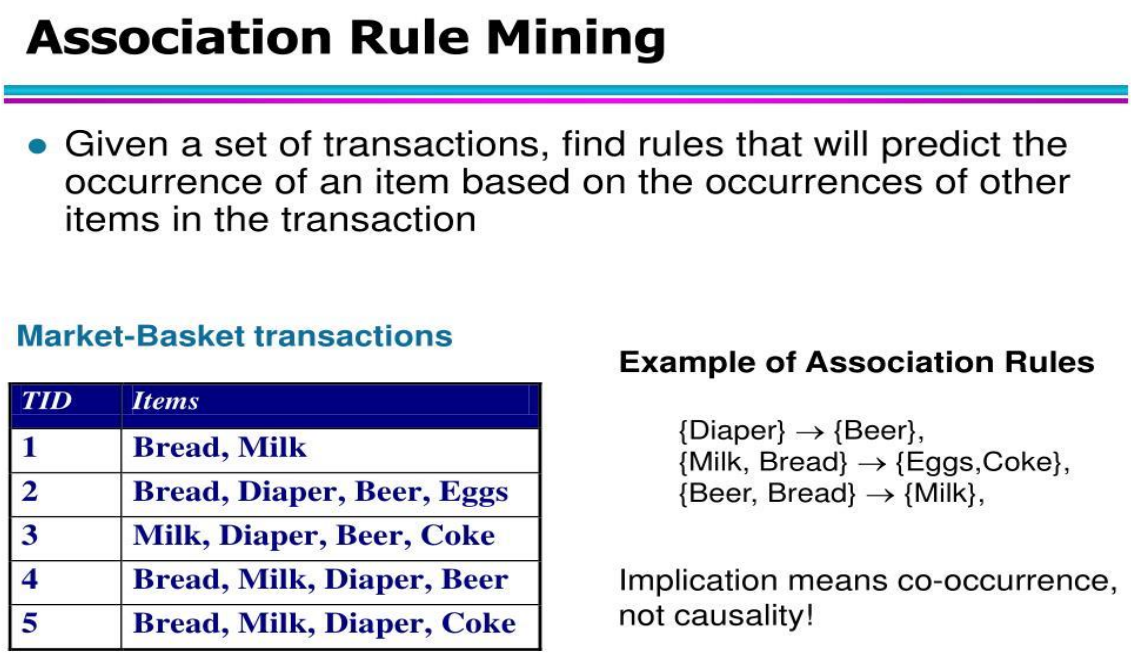
support(A ∪ B) /support(A) =

support count(A ∪ B) / support count(A)

Generally, Association Rule Mining can be viewed in a two-step process:- 1. Find all Frequent itemsets: By definition, each of these itemsets will occur at least as frequently as a pre-established minimum support count, min sup. 2. Generate Association Rules from the Frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

Association Rule Mining :

It is primarily used when you want to identify an association between different items in a set, then find frequent patterns in a transactional database, relational databases(RDBMS). The best example of the association is as you can see in the following image.



Algorithms used in Market Basket Analysis

There are Multiple Techniques and Algorithms are used in Market Basket Analysis. One of the important objectives is “to predict the probability of items that are being bought together by customers”.

- AIS
- SETM Algorithm
- Apriori Algorithm
- FP Growth

- **Apriori Algorithm:**

Apriori Algorithm is a widely-used and well-known Association Rule algorithm and is a popular algorithm used in market basket analysis. It is also considered accurate and overtop AIS and SETM algorithms. It helps to find frequent itemsets in transactions and identifies association rules between these items. The limitation of the Apriori Algorithm is frequent itemset generation. It needs to scan the database many times which leads to increased time and reduce performance as it is a computationally costly step because of a huge database. It uses the concept of Confidence, Support.

- **AIS Algorithm:**

The AIS algorithm creates multiple passes on the entire database or transactional data. During every pass, it scans all transactions. As you can see, in the first pass, it counts the support of separate items and determines then which of them are frequent in the database. Huge itemsets of every pass are enlarged to generate candidate itemsets. After each scanning of a transaction, the common itemsets between these itemsets of the previous pass and then items of this transaction are determined. This algorithm was the first published algorithm which is developed to generate all large itemsets in a transactional database. It was focusing on the enhancement of databases with the necessary performance to process decision support. This technique is bounded to only one item in the consequent.

Advantage:

The AIS algorithm was used to find whether there was an association between items or not.

Disadvantage:

The main disadvantage of the AIS algorithm is that it generates too many candidates set that after turn out to be small. As well as the data structure is to be maintained.

- **SETM Algorithm:**

This Algorithm is quite similar to the AIS algorithm. The SETM algorithm creates collective passes over the database. As you can see, in the first pass, it counts the support of single items and then determines which of them are frequent in the database. Then, it also generates the candidate itemsets by enlarging large itemsets of the

previous pass. In addition to this, the SETM algorithm recalls the TIDs(transaction ids) of the generating transactions with the candidate itemsets.

Advantage:

While generating candidate itemsets, SETM algorithm arranges candidate itemsets together with the TID(transaction Id) in a sequential manner.

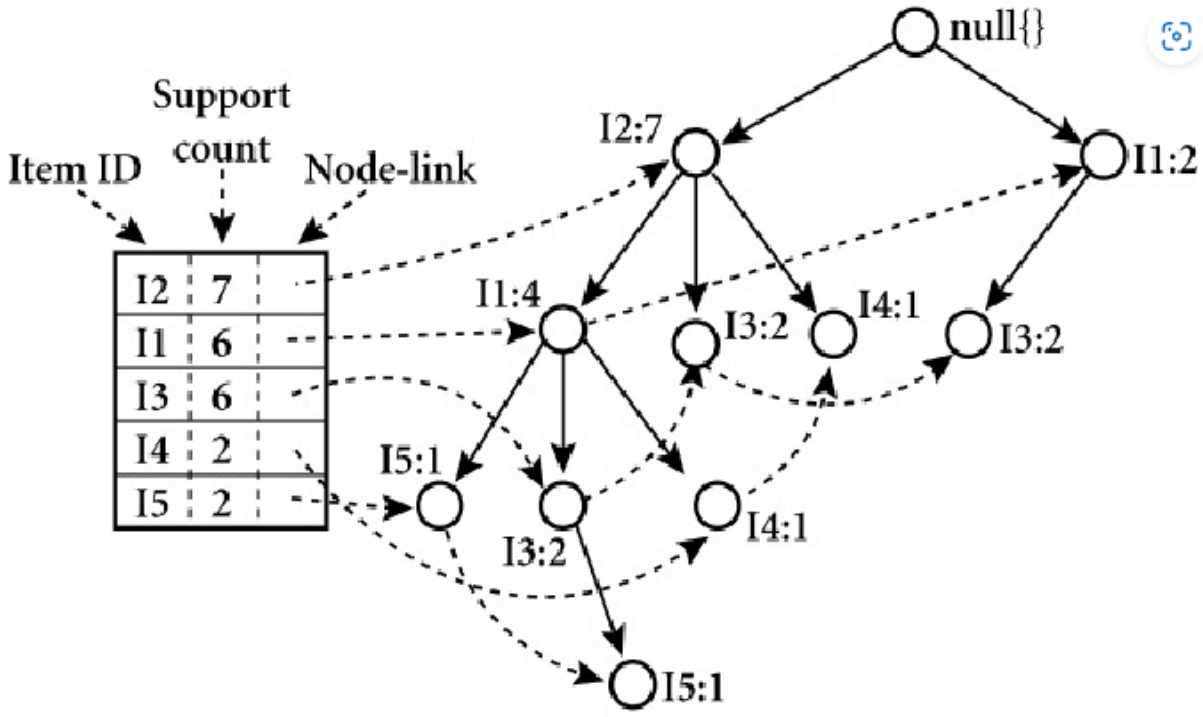
Disadvantage:

For every item set, there is an association with Tid, hence it requires more space to store a huge number of TIDs.

➢ FP Growth:

FP Growth is known as Frequent Pattern Growth Algorithm. FP growth algorithm is a concept of representing the data in the form of an FP tree or Frequent Pattern. Hence FP Growth is a method of Mining Frequent Itemsets. This algorithm is an advancement to the Apriori Algorithm. There is no need for candidate generation to generate the frequent pattern. This frequent pattern tree structure maintains the association between the itemsets. A Frequent Pattern Tree is a tree structure that is made with the earlier itemsets of the data. The main purpose of the FP tree is to mine the most frequent patterns. Every node of the FP tree represents an item of that itemset. The root node represents the null value whereas the lower nodes represent the itemsets of the data. The association of these nodes with the lower nodes that is between itemsets is maintained while creating the tree.

For Example,



Advantages of Market Basket Analysis

There are many advantages to implementing Market Basket Analysis in marketing. Market basket Analysis(MBA) can be applied to data of customers from the point of sale (PoS) systems.

It helps retailers with:

- ✓ Increases customer engagement
- ✓ Boosting sales and increasing RoI
- ✓ Improving customer experience
- ✓ Optimize marketing strategies and campaigns
- ✓ Help to understand customers better
- ✓ Identifies customer behavior and pattern

How does Market Basket Analysis look from the Customer's perspective?

Let us take an example from Amazon, the world's largest eCommerce platform. From a customer's perspective, Market Basket Analysis is like shopping at a supermarket. Generally, it observes all items bought by customers together in a single purchase. Then it shows the most related products together that customers will tend to buy in one purchase

Conclusion:

In conclusion, market basket analysis is a powerful tool for businesses to gain valuable insights into customer behavior and optimize their strategies accordingly

Market Basket Insights - Phase 3 Development Phase Loading and Prepocessing The Transaction Data

The Market Basket Insights project aims to provide valuable insights from transaction data, helping businesses make data-driven decisions. This documentation outlines the key steps involved in the project, starting with the loading and preprocessing of transaction data.

Market Basket Insights is a data analytics project designed to analyze transaction data, often found in retail and e-commerce businesses. The project focuses on understanding customer purchasing behavior and deriving actionable insights from it.

In [ ]:
```
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

Data Loading:

- Begin by obtaining the transaction data, which typically includes information about customer purchases, such as product IDs, quantities, and timestamps.
- Data can be sourced from various platforms and formats, such as databases, spreadsheets, or APIs.

In [ ]:
```
df=pd.read_csv('/kaggle/input/market-basket-analysis/Assignment-1_Data.csv',delimiter=';')
df.head()
```

Out[ ]:

	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

In [ ]:
```
df.info()
```

```
<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
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
df.loc[df['Quantity']<=0][:5]
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
2359	536589	NaN	-10	01.12.2010 16:50	0	NaN	United Kingdom
4289	536764	NaN	-38	02.12.2010 14:42	0	NaN	United Kingdom
6998	536996	NaN	-20	03.12.2010 15:30	0	NaN	United Kingdom
6999	536997	NaN	-20	03.12.2010 15:30	0	NaN	United Kingdom
7000	536998	NaN	-6	03.12.2010 15:30	0	NaN	United Kingdom

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

1-3. | Dropping data with `zero price`

In [ ]:
```
df.loc[df['Price']<='0'][:5
                         ]
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
613	536414	NaN	56	01.12.2010 11:52	0	NaN	United Kingdom
1937	536545	NaN	1	01.12.2010 14:32	0	NaN	United Kingdom
1938	536546	NaN	1	01.12.2010 14:33	0	NaN	United Kingdom
1939	536547	NaN	1	01.12.2010 14:33	0	NaN	United Kingdom
1940	536549	NaN	1	01.12.2010 14:34	0	NaN	United Kingdom

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

1-4. | Dropping `Non-product` data.

In [ ]:
```
df.loc[(df['Itemname']=='POSTAGE')|(df['Itemname']=='DOTCOM POSTAGE')|(df['Itemname']=='Adjust bad debt')|(df['Itemname']=='Manual')]
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
45	536370	POSTAGE	3	01.12.2010 08:45	18	12583.0	France
377	536403	POSTAGE	1	01.12.2010 11:27	15	12791.0	Netherlands
1113	536527	POSTAGE	1	01.12.2010 13:04	18	12662.0	Germany
1781	536544	DOTCOM POSTAGE	1	01.12.2010 14:32	569,77	NaN	United Kingdom
2192	536569	Manual	1	01.12.2010 15:35	1,25	16274.0	United Kingdom

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

1-5. | Filling `null` data

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
BillNo             0
Itemname           0
Quantity           0
Date               0
Price              0
CustomerID    130813
Country            0
dtype: int64
```

In [ ]:
```
df=df.fillna('-')
df.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
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[ ]:

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

Out[ ]:

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

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

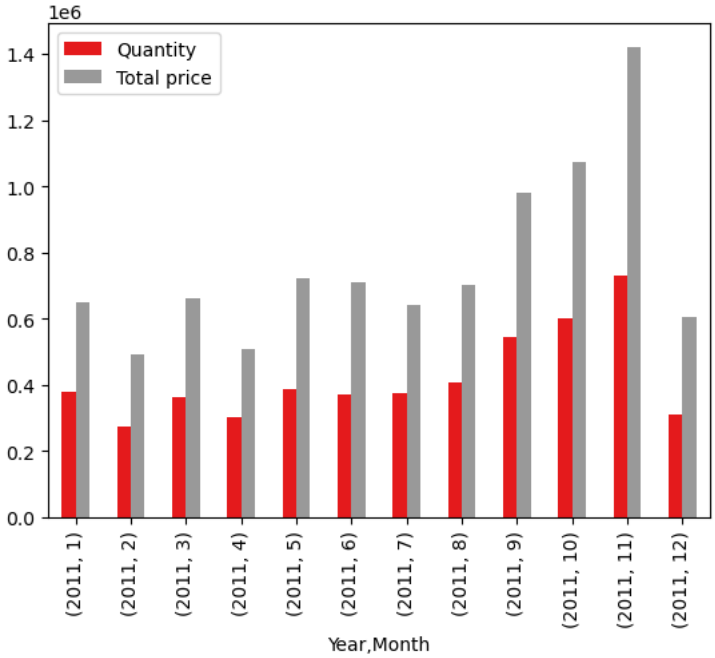
2. | Exploratoty Data Analysis

Exploratory Data Analysis (EDA):

- Conduct an initial exploration of the data to understand its characteristics and identify potential insights.
- EDA may involve generating summary statistics, visualizing item frequencies, and calculating association metrics (e.g., support, confidence, lift).

2-1. | Sales amount and quantity

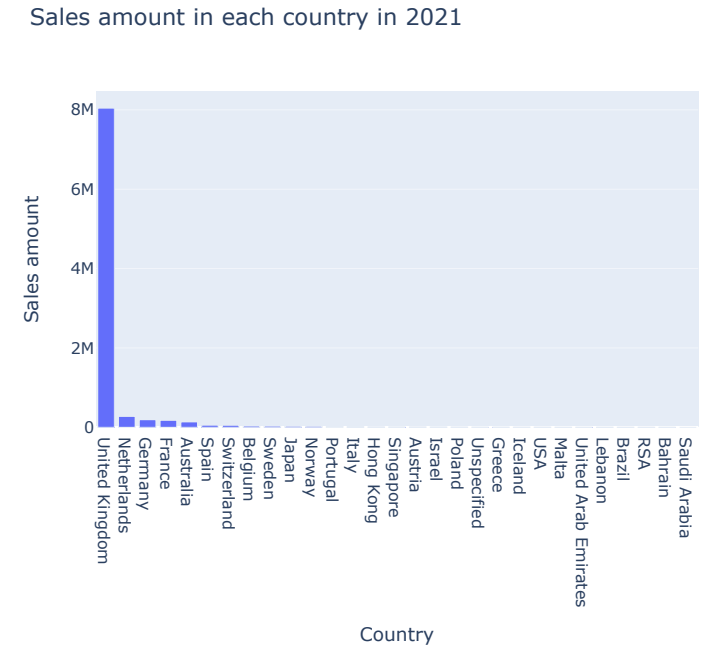
In [ ]:
```
sales=df.groupby(['Year','Month'])['Total price','Quantity'].sum()
sales.to_csv('sales.csv')
sales=pd.read_csv('sales.csv')
sales=sales.pivot_table(sales,index=['Year','Month'],aggfunc=np.sum,fill_value=0)
sales.plot(kind='bar',cmap='Set1')
plt.show()
```



In [ ]:
```
sales_country=df.groupby(['Year','Month','Country'])['Total price'].sum()
sales_country.to_csv('sales_country.csv')
sales_country=pd.read_csv('sales_country.csv')

fig=px.bar(sales_country,x=['Month'],y='Total price',color='Country',title='Monthly sales amount in each country in 2021')
fig.update_layout(xaxis_title='Month',yaxis_title='Sales amount')
fig.show()
```

Most of the sales amounts are occupied by the UK.

In [ ]:
```
country=df.groupby('Country')['Total price'].sum()
country.to_csv('country.csv')
country=pd.read_csv('country.csv')

fig=px.bar(country,x='Country',y='Total price',title='Sales amount in each country in 2021')
fig.update_layout(xaxis={'categoryorder':'total descending'},yaxis_title='Sales amount')
fig.show()
```



2-2. | Category

Top 10 highest sales amount items

In [ ]:
```
cm=sns.light_palette("green",as_cmap=True)

item_sales=df.groupby('Itemname')['Price'].sum().sort_values(ascending=False)[:10]
item_sales.to_csv('item_sales.csv')
```

```
item_sales=pd.read_csv('item_sales.csv')
item_sales.style.background_gradient(cmap=cm).set_precision(2)
```

Out[ ]:

	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

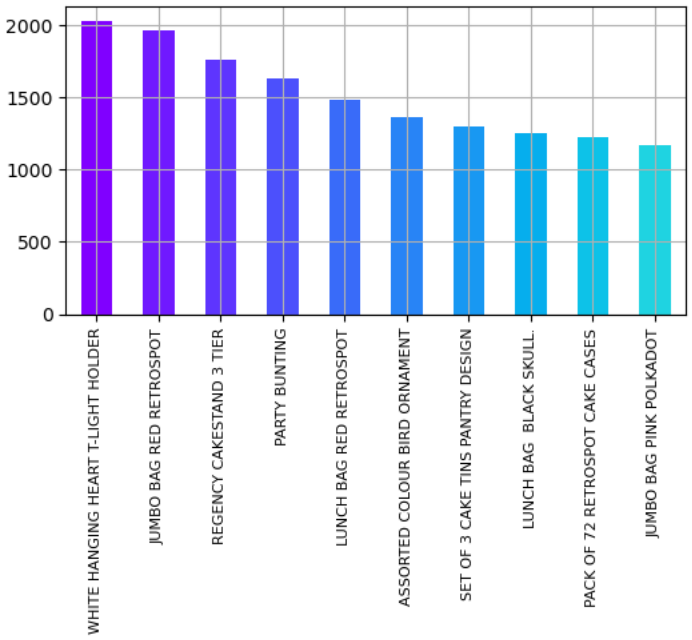
In [ ]:
```
df[['Itemname','Quantity']].sort_values(by='Quantity',ascending=False)[:10].style.background_gradient(cmap=cm).set_precision(2)
```

Out[ ]:

	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 [ ]:
```
color=plt.cm.rainbow(np.linspace(0,1,30))
df['Itemname'].value_counts().head(10).plot.bar(color=color,figsize=(6,3))
# plt.title('Frequency of Most popular items',fontsize=14)
plt.xticks(rotation=90,fontsize=8)
plt.grid()
plt.show()
```



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

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

In [ ]:
```
df['Itemname']=df['Itemname'].str.strip()
df['BillNo']=df['BillNo'].astype('str')
```

In [ ]:
```
basket=(df[df['Country']=='United Kingdom']
        .groupby(['BillNo','Itemname'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('BillNo'))
```

In [ ]:
```
basket.head(3)
```

Out[ ]:

Itemname	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 DAISY PEGS IN WOOD BOX	12 EGG HOUSE PAINTED WOOD	12 HANGING EGGS HAND PAINTED	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	...	ZINC STAR T-LIGHT HOLDER	SWEE SO
BillNo													
539993	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
540001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
540002	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

3 rows × 3893 columns

In [ ]:
```
def encode_units(x):
  if x<=0:
    return 0
  if x>=1:
    return 1
```

In [ ]:
```
basket_sets=basket.applymap(encode_units)
```

In [ ]:
```
frequent_itemsets=apriori(basket_sets,min_support=0.03,use_colnames=True)
```

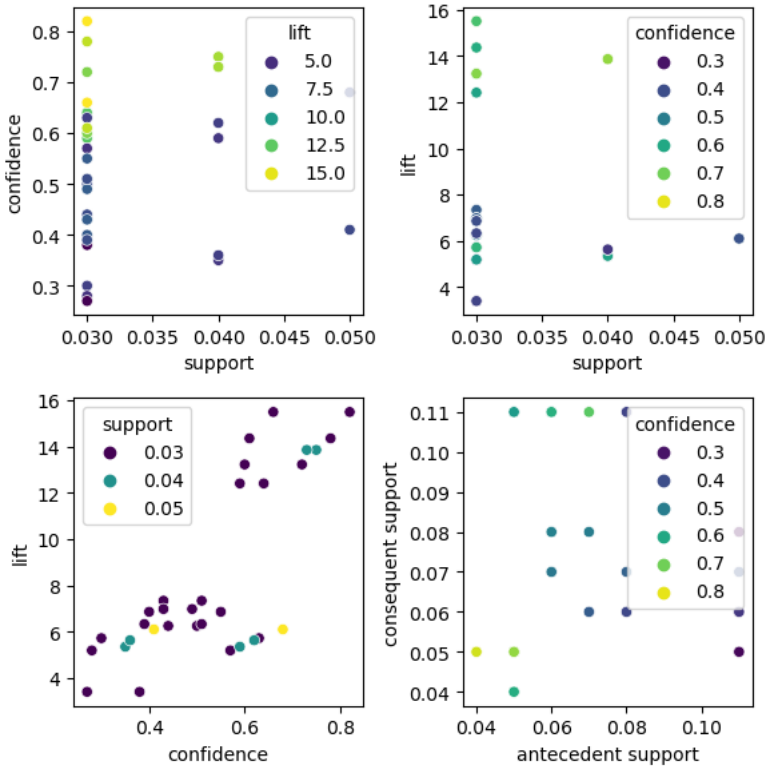
```
/opt/conda/lib/python3.10/site-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning:

DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please u
se a DataFrame with bool type
```

In [ ]:
```
rules=round(association_rules(frequent_itemsets,metric='lift',min_threshold=1),2)
rules.head(5)
```

Out[ ]:

	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 KNEELING PAD KEEP CALM)	(GARDENERS KNEELING PAD CUP OF TEA)	0.05	0.05	0.03	0.60	13.23	0.03	2.40	0.98
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

In [ ]:
```
plt.figure(figsize=(6,6))
plt.subplot(221)
sns.scatterplot(x="support",y="confidence",data=rules,hue="lift",palette="viridis")
plt.subplot(222)
sns.scatterplot(x="support",y="lift",data=rules,hue="confidence",palette="viridis")
plt.subplot(223)
sns.scatterplot(x="confidence",y="lift",data=rules,hue='support',palette="viridis")
plt.subplot(224)
sns.scatterplot(x="antecedent support",y="consequent support",data=rules,hue='confidence',palette="viridis")
plt.tight_layout()
plt.show()
```



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

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

In [ ]:
```
rules[['antecedents','consequents','support']].sort_values('support',ascending=False)[:5].style.background_gradient(cmap=cm).set_prec
```

Out[ ]:

	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*

In [ ]:
```
rules[['antecedents','consequents','confidence']].sort_values('confidence',ascending=False)[:5].style.background_gradient(cmap=cm).se
```

Out[ ]:

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

Out[ ]:

	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

In [ ]:
```
rules[(rules['lift']>=13)&(rules['confidence']>=0.7)].sort_values('lift',ascending=False).style.background_gradient(cmap=cm).set_prec
```

Out[ ]:

	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({'PINK REGENCY TEACUP AND SAUCER'})	frozenset({'ROSES REGENCY TEACUP AND SAUCER'})	0.04	0.05	0.03	0.78	14.36	0.03	4.24	0.97
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

Market Basket Insights is a powerful tool for businesses to gain a deeper understanding of customer behavior and enhance their strategic decisions. By following the steps outlined in this documentation, you can unlock valuable insights from transaction data. Possible future enhancements to the project may include machine learning models for demand forecasting, real-time analytics, and personalization of marketing campaigns.

Market Basket Insights

Phase 4 Development Phase 2 - Performing Different Activities Like Featuer Engineering ,Model training,Evaluation etc.,

Introduction:

Welcome to this notebook where we will explore the fascinating world of Market Basket Analysis using a real-world dataset. Market Basket Analysis is a powerful technique that allows us to uncover patterns and associations between items that customers tend to purchase together. By analyzing these patterns, we can gain valuable insights that can drive business decisions and strategies.

In this notebook, we will work with a Market Basket dataset that captures customer transactions in a retail or e-commerce setting. The dataset provides a wealth of information about customer purchases, allowing us to dive deep into their buying behavior. By leveraging data mining techniques and association rule mining algorithms, we will unravel the relationships between items and discover interesting patterns.

Through this analysis, we can derive actionable insights to improve various aspects of business operations. We can identify frequently co-purchased items, enabling us to make targeted product recommendations and enhance cross-selling and upselling opportunities. By optimizing product placement and store layout based on association patterns, we can create more enticing shopping experiences. Furthermore, we can design effective promotional campaigns by leveraging the discovered item associations, resulting in higher customer engagement and increased sales.

In this notebook, we will take you through the entire process of Market Basket Analysis, from data preprocessing to association rule mining and visualization. By following along with the provided code and explanations, you will gain a solid understanding of how to extract valuable insights from Market Basket datasets and apply them to real-world scenarios.

So let's dive in and unlock the secrets hidden within the Market Basket dataset to gain a deeper understanding of customer behavior and optimize business strategies!

Overview of the Market Basket Analysis dataset

This dataset contains 522,065 rows and 7 attributes that provide valuable information about customer transactions and product details. Here is a breakdown of the attributes:

BillNo: This attribute represents a 6-digit number assigned to each transaction. It serves as a unique identifier for identifying individual purchases.

Itemname: This attribute stores the name of the product purchased in each transaction. It provides nominal data representing different products.

Quantity: This attribute captures the quantity of each product purchased in a transaction. It is a numeric value that indicates the number of units of a specific item.

Date: The Date attribute records the day and time when each transaction occurred. It provides valuable information about the timing of purchases.

Price: This attribute represents the price of each product. It is a numeric value that indicates the cost of a single unit of the item.

CustomerID: Each customer is assigned a 5-digit number as their unique identifier. This attribute helps track customer-specific information and analyze individual buying patterns.

Country: The Country attribute denotes the name of the country where each customer resides. It provides nominal data representing different geographic regions.

By analyzing this dataset, we can gain insights into customer purchasing behavior, identify popular products, examine sales trends over time, and explore the impact of factors such as price and geography on customer preferences. These insights can be used to optimize marketing strategies, improve inventory management, and enhance customer satisfaction.

Data Preprocessing

Importing Required Libraries

In [ ]:
```
import numpy as np  # Import numpy library for efficient array operations
import pandas as pd  # Import pandas library for data processing
import matplotlib.pyplot as plt  # Import matplotlib.pyplot for data visualization
```

Data Loading

Retrieving and Loading the Dataset

In [ ]:
```
df = pd.read_csv('../input/market-basket-analysis/Assignment-1_Data.csv', sep=';',p
df.head()
```

```
/tmp/ipykernel_20/3267796271.py:1: DtypeWarning: Columns (0) have mixed types. Speci
fy dtype option on import or set low_memory=False.
  df = pd.read_csv('../input/market-basket-analysis/Assignment-1_Data.csv', sep=';',
parse_dates=['Date'])
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-01-12 08:26:00	2,55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	2010-01-12 08:26:00	3,39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	2010-01-12 08:26:00	2,75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-01-12 08:26:00	3,39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	2010-01-12 08:26:00	3,39	17850.0	United Kingdom

In [ ]:
```
# Convert the 'Price' column to float64 data type after replacing commas with dots
df['Price'] = df['Price'].str.replace(',', '.').astype('float64')
```

In [ ]:
```
# Display the information about the DataFrame which is to provide an overview of th
df.info()
```

```
<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
```

In [ ]:
```
# Calculate the number of missing values for each column and sort them in descendin
df.isna().sum().sort_values(ascending=False)
```

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

In [ ]:
```
# Calculate the total price by multiplying the quantity and price columns
df['Total_Price'] = df.Quantity * df.Price
```

In [ ]:
```
df.describe(include='all')
```

```
/tmp/ipykernel_20/2884002236.py:1: FutureWarning: Treating datetime data as categori
cal rather than numeric in `.describe` is deprecated and will be removed in a future
version of pandas. Specify `datetime_is_numeric=True` to silence this warning and ad
opt the future behavior now.
  df.describe(include='all')
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Count
count	522064.0	520609	522064.000000	522064	522064.000000	388023.000000	5220
unique	21665.0	4185	NaN	19641	NaN	NaN	
top	573585.0	WHITE HANGING HEART T-LIGHT HOLDER	NaN	2011-10-31 14:41:00	NaN	NaN	Unit Kingdo
freq	1114.0	2269	NaN	1114	NaN	NaN	4876
first	NaN	NaN	NaN	2010-01-12 08:26:00	NaN	NaN	N
last	NaN	NaN	NaN	2011-12-10 17:19:00	NaN	NaN	N
mean	NaN	NaN	10.090435	NaN	3.826801	15316.931710	N
std	NaN	NaN	161.110525	NaN	41.900599	1721.846964	N
min	NaN	NaN	-9600.000000	NaN	-11062.060000	12346.000000	N
25%	NaN	NaN	1.000000	NaN	1.250000	13950.000000	N
50%	NaN	NaN	3.000000	NaN	2.080000	15265.000000	N
75%	NaN	NaN	10.000000	NaN	4.130000	16837.000000	N
max	NaN	NaN	80995.000000	NaN	13541.330000	18287.000000	N

In [ ]:
```
# Print the number of unique countries in the 'Country' column
print("Number of unique countries:", df['Country'].nunique())

# Calculate and print the normalized value counts of the top 5 countries in the 'Co
print(df['Country'].value_counts(normalize=True)[:5])
```

```
Number of unique countries: 30
United Kingdom    0.934027
Germany           0.017320
France            0.016105
Spain             0.004760
Netherlands       0.004526
Name: Country, dtype: float64
```

Considering that the majority of transactions (approximately 93%) in the dataset originate from the UK, the 'Country' column may not contribute significant diversity or variability to the analysis. Therefore, we can choose to remove the 'Country' column from the DataFrame df. we indicate that we want to drop a column, This step allows us to focus on other attributes that may provide more valuable insights for our analysis.

In [ ]:
```
# Delete the 'Country' column from the DataFrame
df.drop('Country', axis=1, inplace=True)
```

In [ ]:
```
# Filter the DataFrame to display rows where 'BillNo' column contains non-digit val
df[df['BillNo'].str.isdigit() == False]
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
288772	A563185	Adjust bad debt	1	2011-12-08 14:50:00	11062.06	NaN	11062.06
288773	A563186	Adjust bad debt	1	2011-12-08 14:51:00	-11062.06	NaN	-11062.06
288774	A563187	Adjust bad debt	1	2011-12-08 14:52:00	-11062.06	NaN	-11062.06

Since the item name "Adjust bad debt" was filled accidentally and does not provide any useful information for our analysis, we can choose to remove the corresponding rows from the DataFrame. The code snippet above filters the DataFrame df to retain only the rows where the 'Itemname' column does not contain the value "Adjust bad debt". This operation effectively eliminates the rows associated with the accidental data entry, ensuring the dataset is free from this irrelevant item name.

In [ ]:
```
# Remove rows where the 'Itemname' column contains "Adjust bad debt"
df = df[df['Itemname'] != "Adjust bad debt"]
```

In [ ]:
```
# Here to check if all BillNo doesn't inculde letters
df['BillNo'].astype("int64")
```

Out[ ]:
```
0         536365
1         536365
2         536365
3         536365
4         536365
           ...
522059    581587
522060    581587
522061    581587
522062    581587
522063    581587
Name: BillNo, Length: 522061, dtype: int64
```

In [ ]:
```
# Calculate the sum of 'Price' for rows where 'Itemname' is missing
df[df['Itemname'].isna()] ['Price'].sum()
```

Out[ ]:
```
0.0
```

Exploring Rows with Missing Item Names:

To investigate the data where the 'Itemname' column has missing values, we can filter the dataset to display only those rows. This subset of the data will provide insights into the records where the item names are not available.

In [ ]:
```
# Filter the DataFrame to display rows where 'Itemname' is missing
df[df['Itemname'].isna()]
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
613	536414	NaN	56	2010-01-12 11:52:00	0.0	NaN	0.0
1937	536545	NaN	1	2010-01-12 14:32:00	0.0	NaN	0.0
1938	536546	NaN	1	2010-01-12 14:33:00	0.0	NaN	0.0
1939	536547	NaN	1	2010-01-12 14:33:00	0.0	NaN	0.0
1940	536549	NaN	1	2010-01-12 14:34:00	0.0	NaN	0.0
...	...	...	...	...	...	...	...
515623	581199	NaN	-2	2011-07-12 18:26:00	0.0	NaN	-0.0
515627	581203	NaN	15	2011-07-12 18:31:00	0.0	NaN	0.0
515633	581209	NaN	6	2011-07-12 18:35:00	0.0	NaN	0.0
517266	581234	NaN	27	2011-08-12 10:33:00	0.0	NaN	0.0
518820	581408	NaN	20	2011-08-12 14:06:00	0.0	NaN	0.0

1455 rows × 7 columns

Upon examining the data where the 'Itemname' column has missing values, it becomes evident that these missing entries do not contribute any meaningful information. Given that the item names are not available for these records, it suggests that these instances may not be crucial for our analysis. As a result, we can consider these missing values as non-significant and proceed with our analysis without incorporating them.

In [ ]:

```
# Filter the DataFrame to exclude rows where 'Itemname' is missing (not NaN)
df = df[df['Itemname'].notna()]

# Print the number of unique items in the 'Itemname' column
print("Number of unique items:", df['Itemname'].nunique())

# Calculate and print the normalized value counts of the top 5 items in the 'Itemna
print(df['Itemname'].value_counts(normalize=True)[:5])
```

```
Number of unique items: 4184
WHITE HANGING HEART T-LIGHT HOLDER    0.004358
JUMBO BAG RED RETROSPOT               0.004009
REGENCY CAKESTAND 3 TIER              0.003707
PARTY BUNTING                         0.003221
LUNCH BAG RED RETROSPOT               0.003016
Name: Itemname, dtype: float64
```

A curious observation has caught our attention—the presence of a negative quantity in the 515,623rd row.

we are intrigued by the existence of negative quantities within the dataset. To gain a deeper understanding of this phenomenon, we focus our attention on these specific instances and aim to uncover the underlying reasons behind their occurrence. Through this exploration, we expect to gain valuable insights into the nature of these negative quantities and their potential impact on our analysis. Our investigation aims to reveal the intriguing stories that lie within this aspect of the data.

In [ ]:
```
# Filter the DataFrame to display rows where 'Quantity' is less than 1
df[df['Quantity'] < 1]
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
7122	537032	?	-30	2010-03-12 16:50:00	0.0	NaN	-0.0
12926	537425	check	-20	2010-06-12 15:35:00	0.0	NaN	-0.0
12927	537426	check	-35	2010-06-12 15:36:00	0.0	NaN	-0.0
12973	537432	damages	-43	2010-06-12 16:10:00	0.0	NaN	-0.0
20844	538072	faulty	-13	2010-09-12 14:10:00	0.0	NaN	-0.0
...	...	...	...	...	...	...	...
515634	581210	check	-26	2011-07-12 18:36:00	0.0	NaN	-0.0
515636	581212	lost	-1050	2011-07-12 18:38:00	0.0	NaN	-0.0
515637	581213	check	-30	2011-07-12 18:38:00	0.0	NaN	-0.0
517209	581226	missing	-338	2011-08-12 09:56:00	0.0	NaN	-0.0
519172	581422	smashed	-235	2011-08-12 15:24:00	0.0	NaN	-0.0

473 rows × 7 columns

Given the observation that negative quantities might be filled with system issues or irrelevant information for our analysis, it is reasonable to proceed with removing these rows from the dataset. By doing so, we can ensure the accuracy and reliability of our data, as well as eliminate potential biases or misleading information stemming from negative quantities.

In [ ]:
```
# Remove rows where 'Quantity' is less than 1
df = df[df['Quantity'] >= 1]
```

Next, we turn our attention to the presence of missing values in the 'CustomerID' column. By investigating these missing values, we aim to identify any potential issues or data quality concerns associated with them. Analyzing the impact of missing 'CustomerID' values will help us assess the completeness and reliability of the dataset, enabling us to make informed decisions on handling or imputing these missing values. Let's dive deeper into this aspect and gain a comprehensive understanding of any issues related to missing 'CustomerID' values.

In [ ]:
```
# Select a random sample of 30 rows where 'CustomerID' is missing
df[df['CustomerID'].isna()].sample(30)
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
214695	556428	JUMBO BAG PINK VINTAGE PAISLEY	5	2011-10-06 13:23:00	4.13	NaN	20.65
107468	545714	SET/20 RED RETROSPOT PAPER NAPKINS	2	2011-07-03 10:12:00	2.46	NaN	4.92
502141	580367	TROPICAL PASSPORT COVER	1	2011-02-12 16:39:00	4.13	NaN	4.13
483171	578833	NATURAL SLATE HEART CHALKBOARD	2	2011-11-25 15:23:00	5.79	NaN	11.58
141679	549023	JAM MAKING SET PRINTED	2	2011-05-04 16:27:00	3.29	NaN	6.58
418976	574076	SPACEBOY CHILDRENS BOWL	1	2011-02-11 15:38:00	2.46	NaN	2.46
286153	562934	SET/3 RED GINGHAM ROSE STORAGE BOX	1	2011-10-08 16:56:00	7.46	NaN	7.46
65382	541827	AIRLINE BAG VINTAGE WORLD CHAMPION	2	2011-01-21 17:05:00	4.13	NaN	8.26
363110	569545	ANGEL DECORATION PAINTED ZINC	5	2011-04-10 16:37:00	0.63	NaN	3.15
442144	575875	SET OF 3 WOODEN TREE DECORATIONS	2	2011-11-11 13:06:00	2.46	NaN	4.92
495432	579694	CERAMIC HEART FAIRY CAKE MONEY BANK	4	2011-11-30 14:11:00	4.13	NaN	16.52
118295	546884	SMALL RED RETROSPOT MUG IN BOX	1	2011-03-17 18:08:00	7.46	NaN	7.46
15346	537640	FAWN BLUE HOT WATER BOTTLE	33	2010-07-12 15:31:00	6.77	NaN	223.41
23891	538349	CAKE STAND VICTORIAN FILIGREE SMALL	1	2010-10-12 14:59:00	4.21	NaN	4.21

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
414698	573585	BLACK 3 BEAD DROP EARRINGS	1	2011-10-31 14:41:00	2.90	NaN	2.90
64071	541696	CARROT CHARLIE+LOLA COASTER SET	1	2011-01-20 18:08:00	3.29	NaN	3.29
483891	578844	CHARLIE+LOLA PINK HOT WATER BOTTLE	2	2011-11-25 16:14:00	3.29	NaN	6.58
442218	575875	LUNCH BAG RED VINTAGE DOILY	1	2011-11-11 13:06:00	4.13	NaN	4.13
60605	541497	RED FLORAL FELTCRAFT SHOULDER BAG	1	2011-01-18 15:19:00	7.46	NaN	7.46
67706	541971	BIRD DECORATION RED RETROSPOT	1	2011-01-24 13:48:00	1.63	NaN	1.63
37046	539492	CERAMIC LOVE HEART MONEY BANK	1	2010-12-20 10:14:00	3.36	NaN	3.36
53991	540995	GREEN ROSE WASHBAG	2	2011-01-13 09:30:00	3.29	NaN	6.58
151970	550208	SEWING SUSAN 21 NEEDLE SET	1	2011-04-15 10:39:00	1.63	NaN	1.63
95503	544684	SET OF 4 ENGLISH ROSE COASTERS	2	2011-02-22 16:32:00	2.46	NaN	4.92
418793	574076	FELT EGG COSY CHICKEN	1	2011-02-11 15:38:00	1.63	NaN	1.63
477333	578344	ASSORTED TUTTI FRUTTI BRACELET	10	2011-11-24 09:21:00	0.83	NaN	8.30
317687	565917	SET 12 KIDS COLOUR CHALK STICKS	1	2011-07-09 16:15:00	0.83	NaN	0.83
520696	581492	FEATHER PEN,LIGHT PINK	1	2011-09-12 10:03:00	0.83	NaN	0.83

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
286563	562955	TRAVEL CARD WALLET SUKI	3	2011-11-08 10:14:00	0.83	NaN	2.49
239479	558837	TOY TIDY PINK POLKADOT	1	2011-04-07 11:58:00	4.96	NaN	4.96

This sample can provide us with a glimpse into the specific instances where 'CustomerID' is missing, aiding us in further analysis or decision-making related to handling these missing values.

Upon analyzing a sample of rows where the 'CustomerID' is missing, it appears that there is no discernible pattern or specific reason behind the absence of these values. This observation suggests that the missing 'CustomerID' entries were not filled accidentally or due to a systematic issue. Instead, it is possible that these missing values occur naturally in the dataset, without any particular significance or underlying cause.

Identifying Issues in the Price Column: Ensuring Data Quality

In our analysis, we shift our focus to the 'Price' column and investigate it for any potential issues or anomalies. By thoroughly examining the data within this column, we aim to identify any irregularities, inconsistencies, or outliers that may affect the overall quality and integrity of the dataset. Analyzing the 'Price' column is crucial in ensuring accurate and reliable pricing information for our analysis. Let's dive deeper into the 'Price' column and uncover any issues that may require attention.

In [ ]:
```
# Counting the number of rows where the price is zero
zero_price_count = len(df[df['Price'] == 0])
print("Number of rows where price is zero:", zero_price_count)

# Counting the number of rows where the price is negative
negative_price_count = len(df[df['Price'] < 0])
print("Number of rows where price is negative:", negative_price_count)
```

```
Number of rows where price is zero: 583
Number of rows where price is negative: 0
```

our attention now turns to the presence of zero charges in the 'Price' column. It is important to explore instances where products were offered free of cost, as this information can provide valuable insights into promotional activities, giveaways, or other unique aspects of the dataset. By examining the data related to zero charges in the 'Price' column, we can gain a deeper understanding of these transactions and their potential impact on our analysis. Let's delve into the details of these zero-priced transactions and uncover any significant findings.

In [ ]:
```
# Selecting a random sample of 20 rows where the price is zero
df[df['Price'] == 0].sample(20)
```

Out[ ]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
407408	573169	Found by jackie	1	2011-10-28 09:56:00	0.0	NaN	0.0
51894	540832	RED RETROSPOT CHARLOTTE BAG	1	2011-11-01 14:42:00	0.0	NaN	0.0
101152	545176	RECIPE BOX BLUE SKETCHBOOK DESIGN	1	2011-02-28 14:19:00	0.0	NaN	0.0
416909	573991	check	36	2011-02-11 11:19:00	0.0	NaN	0.0
186886	553539	CERAMIC STRAWBERRY CAKE MONEY BANK	1	2011-05-17 15:27:00	0.0	NaN	0.0
233712	558340	RECIPE BOX PANTRY YELLOW DESIGN	1	2011-06-28 14:01:00	0.0	NaN	0.0
14051	537534	RED KITCHEN SCALES	1	2010-07-12 11:48:00	0.0	NaN	0.0
233684	558340	CHILDS GARDEN SPADE BLUE	1	2011-06-28 14:01:00	0.0	NaN	0.0
39894	539750	FAWN BLUE HOT WATER BOTTLE	1	2010-12-21 15:40:00	0.0	NaN	0.0
119506	546933	CERAMIC STRAWBERRY MONEY BOX	2	2011-03-18 11:02:00	0.0	NaN	0.0
461978	577263	found	66	2011-11-18 12:31:00	0.0	NaN	0.0
237811	558725	found	37	2011-01-07 14:31:00	0.0	NaN	0.0
14052	537534	IVORY KITCHEN SCALES	3	2010-07-12 11:48:00	0.0	NaN	0.0
518770	581406	POLYESTER FILLER PAD 45x45cm	240	2011-08-12 13:58:00	0.0	NaN	0.0

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
186876	553539	BISCUIT TIN VINTAGE GREEN	1	2011-05-17 15:27:00	0.0	NaN	0.0
101144	545176	DOORMAT MERRY CHRISTMAS RED	3	2011-02-28 14:19:00	0.0	NaN	0.0
14068	537534	AIRLINE BAG VINTAGE JET SET WHITE	1	2010-07-12 11:48:00	0.0	NaN	0.0
301840	564530	FRENCH BLUE METAL DOOR SIGN 8	3	2011-08-25 14:57:00	0.0	NaN	0.0
101148	545176	FRENCH BLUE METAL DOOR SIGN 6	3	2011-02-28 14:19:00	0.0	NaN	0.0
40295	539856	RED RETROSPOT TEA CUP AND SAUCER	1	2010-12-22 14:41:00	0.0	NaN	0.0

Removing Rows with Zero Price: Eliminating Misleading Data Entries

Upon reviewing the sample of rows where the price is zero, we have identified that these entries might provide misleading or inaccurate information for our analysis. Therefore, it is prudent to proceed with removing these rows from the dataset to ensure the integrity and reliability of our analysis.

In [ ]:
```
# Remove rows where the price is zero
df = df[df['Price'] != 0]
```

Data Understanding: Exploring and Interpreting the Dataset

In the data analysis process, data understanding plays a crucial role in gaining insights and formulating meaningful conclusions. By thoroughly examining the dataset, we aim to understand its structure, contents, and underlying patterns. This understanding empowers us to make informed decisions regarding data cleaning, feature engineering, and subsequent analysis steps.

Key aspects of data understanding include:

```
Exploring the Dataset: We investigate the dataset's dimensions,
such as the number of rows and columns, to gauge its size and
complexity. Additionally, we examine the data types of each column
```

to understand the nature of the variables.

Assessing Data Quality: We scrutinize the data for inconsistencies, outliers, or other data quality issues that may require attention. Addressing these issues ensures the reliability and accuracy of the data.

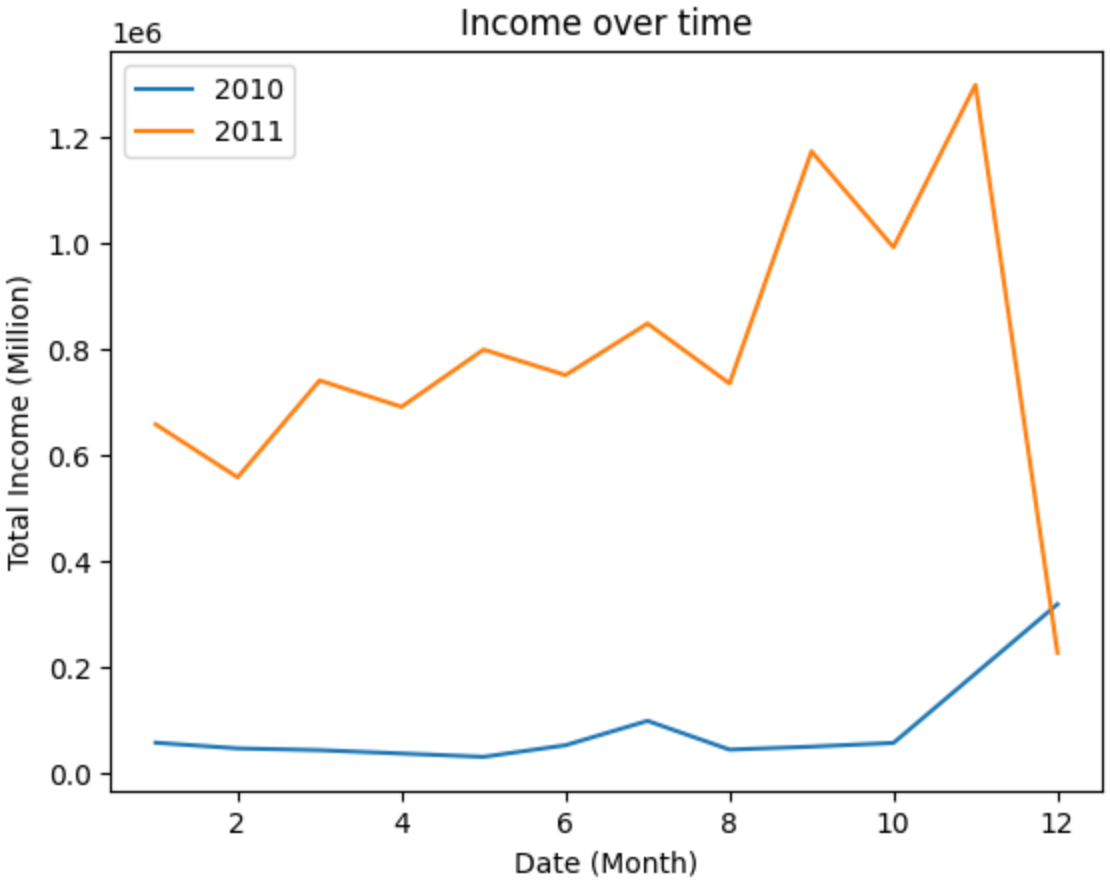
Identifying Relationships: We analyze the relationships between variables by examining correlations, associations, or dependencies. This analysis allows us to uncover meaningful connections that can drive insights and guide our analysis.

Detecting Patterns and Trends: We look for recurring patterns, trends, or distributions within the data. This step can reveal valuable information about customer behavior, market dynamics, or other relevant factors.

By thoroughly understanding the dataset, we lay the foundation for meaningful data analysis and generate insights that contribute to informed decision-making and problem-solving.

In [ ]:
```
# Grouping the data by month and summing the total price for the year 2010
df[df["Date"].dt.year == 2010].groupby(df["Date"].dt.month)["Total_Price"].sum().pl

# Grouping the data by month and summing the total price for the year 2011
df[df["Date"].dt.year == 2011].groupby(df["Date"].dt.month)["Total_Price"].sum().pl

# Adding legend and plot labels
plt.legend(["2010", "2011"])
plt.title("Income over time")
plt.ylabel('Total Income (Million)')
plt.xlabel("Date (Month)")
```

Out[ ]:
```
Text(0.5, 0, 'Date (Month)')
```



The code snippet above creates a line plot to visualize the income over time for the years 2010 and 2011. First, the data is filtered based on the year using the dt.year attribute of the 'Date' column. The data is then grouped by month, and the 'Total_Price' column is summed. Two line plots are created, one for each year, showing the monthly total income. The legend is added to indicate the respective years, and the plot is labeled with a title, y-axis label, and x-axis label. This visualization allows us to observe the trend and compare the income between the two years.

Upon observing the line plot of income over time for the years 2010 and 2011, it becomes apparent that the sales remained relatively stable and consistent until October 2010. This suggests that the business was growing steadily during this period, as the sales continued to increase.

However, a significant drop in sales is observed in the last month of the dataset. This sudden decline indicates a notable deviation from the previously observed growth trend. Exploring the potential factors contributing to this drop becomes crucial in understanding the underlying reasons for the decline in sales during that specific period.

To verify if the data is complete for the entire last month in the dataset, we can compare the maximum date in the 'Date' column with the last day of that month. If they match, it indicates that the data is filled for the entire last month.

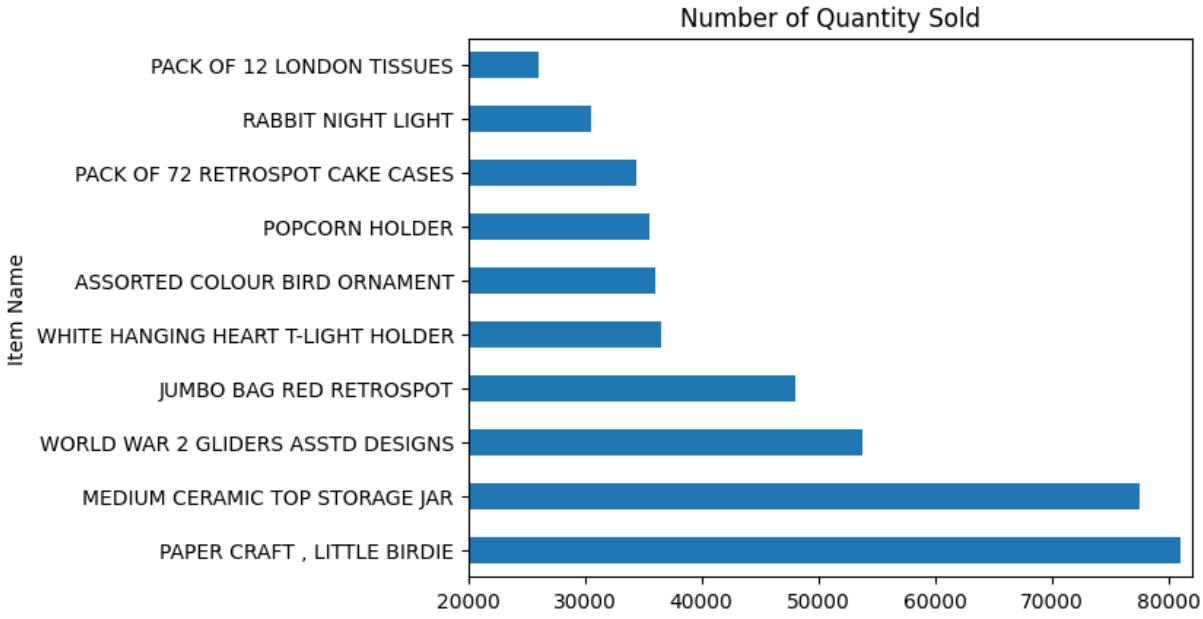
In [ ]:
```
df["Date"].max()
```

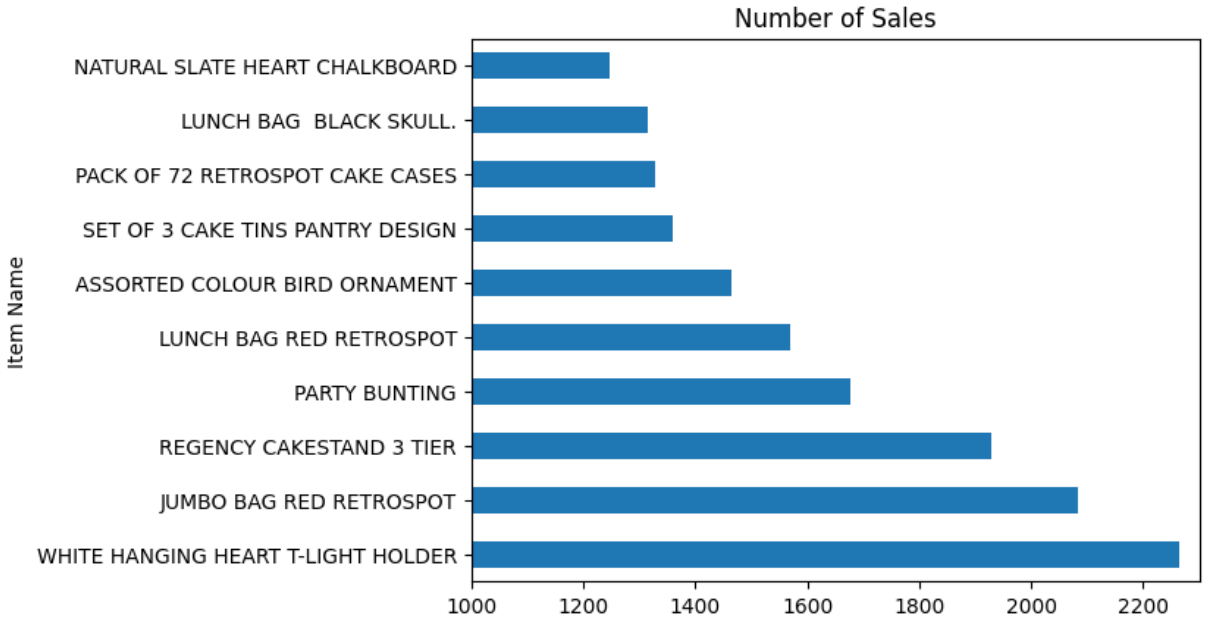
Out[ ]:
```
Timestamp('2011-12-10 17:19:00')
```

Based on the finding that the data is only available for 10 days in the last month, it becomes evident that the significant drop in sales observed during that period is likely due to the limited data rather than an actual decline in sales. The incomplete data for the last month may not provide a comprehensive representation of the sales performance during that period.

To gain a more accurate understanding of the sales trend, it is advisable to consider a broader time frame with complete data. Analyzing a more extended period that encompasses multiple months or years would provide a more reliable assessment of the sales performance and allow for more meaningful insights and conclusions.

In [ ]:
```
# Plotting the top 10 most sold products by quantity
df.groupby('Itemname')['Quantity'].sum().sort_values(ascending=False)[:10].plot(kin
plt.ylabel('Item Name')
plt.xlim(20000, 82000)
plt.show()

# Plotting the top 10 most sold products by count
df['Itemname'].value_counts(ascending=False)[:10].plot(kind='barh', title='Number o
plt.ylabel('Item Name')
plt.xlim(1000, 2300)
plt.show()
```





The code snippet above creates two horizontal bar plots to visualize the most sold products based on quantity and count, respectively.

In the first plot, the top 10 items are determined by summing the 'Quantity' column for each unique 'Itemname' and sorting them in descending order. The plot displays the number of quantities sold for each item.

The second plot showcases the top 10 items based on the count of sales for each unique 'Itemname'. The value_counts function counts the occurrences of each item and sorts them in descending order. The plot represents the number of times each item has been sold.

Observing the plots, we can infer that there are products that are sold more frequently (higher count) compared to others, despite having relatively lower quantities sold per transaction. This indicates the presence of items that are commonly purchased in larger quantities at once. These products might include items that are frequently bought in bulk or items that are typically sold in larger packages or quantities.

This insight highlights the importance of considering both the quantity sold and the count of sales when analyzing the popularity and demand for different products. It suggests that some items may have a higher turnover rate due to frequent purchases, while others may have a higher quantity per sale, leading to different sales patterns and customer behaviors. Understanding these dynamics can be valuable for inventory management, pricing strategies, and identifying customer preferences.

Association Rules

Association rules are generated using the Apriori algorithm, which is a popular algorithm for discovering interesting relationships or associations among items in a dataset. Association

rule mining is commonly used in market basket analysis, where the goal is to find associations between items frequently purchased together.

The generated association rules provide insights into the relationships between different items or itemsets in the dataset. Each association rule consists of two parts: the antecedent (or left-hand side) and the consequent (or right-hand side). The antecedent represents the item(s) or itemset(s) that act as a condition or premise, while the consequent represents the item(s) or itemset(s) that are predicted or inferred from the antecedent.

The association rules are evaluated based on different metrics, such as support, confidence, lift, leverage, and conviction. These metrics provide measures of the interestingness or strength of the rules.

- Support measures the proportion of transactions in the dataset that contain both the antecedent and the consequent.
- Confidence measures the conditional probability of the consequent given the antecedent.
- Lift measures the ratio of observed support to expected support, indicating the strength of the association between the antecedent and the consequent.
- Leverage measures the difference between the observed support and the expected support, indicating the significance of the association.
- Conviction measures the ratio of the expected confidence to the observed confidence, indicating the degree of dependency between the antecedent and the consequent.

By examining the association rules, you can identify interesting relationships, co-occurrences, or patterns among items, which can be used for various purposes such as product recommendation, market segmentation, or inventory management.

To generate the association rules, we use the Apriori algorithm with a minimum support threshold of 0.05 (5%). This ensures that only itemsets with sufficient frequency in the dataset are considered.

Let's explore the generated association rules:

In [ ]:
```
# Assign the original DataFrame to df2
df2 = df

# Filter rows based on item occurrences
item_counts = df2['Itemname'].value_counts(ascending=False)
filtered_items = item_counts.loc[item_counts > 1].reset_index()['index']
df2 = df2[df2['Itemname'].isin(filtered_items)]

# Filter rows based on bill number occurrences
bill_counts = df2['BillNo'].value_counts(ascending=False)
filtered_bills = bill_counts.loc[bill_counts > 1].reset_index()['index']
df2 = df2[df2['BillNo'].isin(filtered_bills)]
```

Filtering is done based on item occurrences:

```
    The frequency count of each unique item name in the 'Itemname'
column is calculated and stored in item_counts.
    filtered_items is created by filtering item_counts to retain
only item names that occur more than once.
    Rows in df2 are filtered to keep only those where the item name
in the 'Itemname' column is present in the filtered_items list.
```

Filtering is done based on bill number occurrences:

```
    The frequency count of each unique bill number in the 'BillNo'
column is calculated and stored in bill_counts.
    filtered_bills is created by filtering bill_counts to retain
only bill numbers that occur more than once.
    Rows in df2 are filtered to keep only those where the bill
number in the 'BillNo' column is present in the filtered_bills
list.
```

After executing the code, the filtered DataFrame df2 will contain only the rows where both the item name and bill number occur more than once in the original df.

In [ ]:
```
# Create a pivot table using the filtered DataFrame
pivot_table = pd.pivot_table(df2[['BillNo','Itemname']], index='BillNo', columns='I
```

The code creates a pivot table that represents the occurrence of items in bills. The pivot table provides a binary representation where each cell indicates whether a specific item appears in a particular bill. Here's how it works:

```
The original DataFrame df2 contains information about bills and
corresponding item names.
By using the pd.pivot_table() function, we reshape the DataFrame to
create a pivot table.
The pivot table has 'BillNo' as the index and 'Itemname' as the
columns, grouping the data based on these two columns.
The goal is to determine whether a specific item appears in a
particular bill.
Each cell in the pivot table is filled with either True or False:
    If an item appears in a bill, the corresponding cell is marked
as True.
    If an item does not appear in a bill, the corresponding cell is
marked as False.
This binary representation of item occurrence in bills allows us to
easily analyze and identify patterns or associations between
different items and bills.
```

The resulting pivot table provides a concise summary of the occurrence of items in bills, which can be used for various purposes such as market basket analysis, recommendation systems, or identifying frequent itemsets and association rules.

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

# Generate frequent itemsets with minimum support of 0.1 (10%)
frequent_itemsets = apriori(pivot_table, min_support=0.01,use_colnames=True)

# Generate association rules
rules = association_rules(frequent_itemsets, "confidence", min_threshold = 0.5)

# Print frequent itemsets
print("Frequent Itemsets:")
print(frequent_itemsets)

# Print association rules
print("\nAssociation Rules:")
rules
```

```
Frequent Itemsets:
       support                                           itemsets
0     0.017370                          (10 COLOUR SPACEBOY PEN)
1     0.013751                 (12 MESSAGE CARDS WITH ENVELOPES)
2     0.019653                   (12 PENCIL SMALL TUBE WOODLAND)
3     0.019820               (12 PENCILS SMALL TUBE RED RETROSPOT)
4     0.019597                     (12 PENCILS SMALL TUBE SKULL)
...        ...                                               ...
2467  0.010355  (LUNCH BAG RED RETROSPOT, LUNCH BAG SUKI DESIG...
2468  0.010188  (LUNCH BAG RED RETROSPOT, LUNCH BAG SUKI DESIG...
2469  0.010300  (LUNCH BAG RED RETROSPOT, LUNCH BAG SPACEBOY D...
2470  0.010467  (LUNCH BAG RED RETROSPOT, LUNCH BAG PINK POLKA...
2471  0.011302  (CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL...

[2472 rows x 2 columns]

Association Rules:
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(60 CAKE CASES DOLLY GIRL DESIGN)	(PACK OF 72 RETROSPOT CAKE CASES)	0.023160	0.071206	0.013028	0.562500	7.899629
1	(60 TEATIME FAIRY CAKE CASES)	(PACK OF 72 RETROSPOT CAKE CASES)	0.044427	0.071206	0.024218	0.545113	7.655446
2	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE GREEN)	0.023216	0.053558	0.015254	0.657074	12.268575
3	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE PINK)	0.023216	0.042256	0.011691	0.503597	11.917802
4	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE RED)	0.023216	0.057121	0.015811	0.681055	11.923112
...	...	...	...	...	...	...	..
1392	(CHARLOTTE BAG SUKI DESIGN, STRAWBERRY CHARLOT...	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	0.018483	0.021824	0.011302	0.611446	28.017319
1393	(CHARLOTTE BAG SUKI DESIGN, WOODLAND CHARLOTTE...	(CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL...	0.018595	0.020989	0.011302	0.607784	28.957623
1394	(CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL...	(CHARLOTTE BAG SUKI DESIGN, WOODLAND CHARLOTTE...	0.020989	0.018595	0.011302	0.538462	28.957623
1395	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	(CHARLOTTE BAG SUKI DESIGN, STRAWBERRY CHARLOT...	0.021824	0.018483	0.011302	0.517857	28.017319
1396	(WOODLAND CHARLOTTE BAG, STRAWBERRY CHARLOTTE ...	(CHARLOTTE BAG PINK POLKADOT, CHARLOTTE BAG SU...	0.022492	0.018261	0.011302	0.502475	27.516648

1397 rows × 10 columns

The code uses the apriori algorithm and association rule mining techniques to analyze the occurrence of items in bills. Here's the overall idea:

```
Frequent Itemsets Generation:
    The apriori algorithm is applied to the pivot_table created
earlier, which represents the occurrence of items in bills.
    The algorithm identifies sets of items that frequently co-occur
together in the bills.
    The minimum support threshold of 0.01 (1%) is set, meaning that
an itemset must occur in at least 1% of the bills to be considered
frequent.
    The resulting frequent itemsets represent combinations of items
that are frequently observed together in bills.

Association Rules Generation:
    Using the frequent itemsets, association rules are generated.
    Association rules capture relationships and patterns between
items based on their co-occurrence in bills.
    The confidence metric is used to evaluate the strength of the
rules. Confidence measures how often the consequent item(s) appear
in bills when the antecedent item(s) are present.
    A minimum confidence threshold of 0.5 (50%) is set, meaning
that only rules with a confidence greater than or equal to 0.5 will
be considered significant.
```

By applying these techniques to the pivot_table, the code enables the discovery of frequent itemsets and the extraction of meaningful association rules, helping to uncover hidden patterns and relationships in the data.

In [ ]:
```
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])

rules
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lif
17	(BEADED CRYSTAL HEART PINK ON STICK)	(DOTCOM POSTAGE)	0.011469	0.039305	0.011190	0.975728	24.82440
614	(HERB MARKER CHIVES, HERB MARKER THYME)	(HERB MARKER PARSLEY)	0.010411	0.012916	0.010077	0.967914	74.93827
607	(HERB MARKER CHIVES, HERB MARKER ROSEMARY)	(HERB MARKER PARSLEY)	0.010355	0.012916	0.010021	0.967742	74.92491
619	(HERB MARKER CHIVES, HERB MARKER ROSEMARY)	(HERB MARKER THYME)	0.010355	0.012916	0.010021	0.967742	74.92491
1217	(HERB MARKER BASIL, HERB MARKER ROSEMARY, HERB...	(HERB MARKER THYME)	0.010578	0.012916	0.010188	0.963158	74.57000
...	...	...	...	...	...	...	..
25	(RED RETROSPOT CUP)	(BLUE POLKADOT CUP)	0.021378	0.018038	0.010689	0.500000	27.71913
1159	(STRAWBERRY CHARLOTTE BAG, RED RETROSPOT CHARL...	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	0.026834	0.021824	0.013417	0.500000	22.91071
113	(HAND WARMER RED LOVE HEART)	(HAND WARMER SCOTTY DOG DESIGN)	0.021935	0.030286	0.010968	0.500000	16.50919
147	(LOVE HOT WATER BOTTLE)	(HOT WATER BOTTLE KEEP CALM)	0.025832	0.042701	0.012916	0.500000	11.70925
370	(CHARLOTTE BAG PINK POLKADOT,	(PACK OF 72 RETROSPOT CAKE CASES)	0.021824	0.071206	0.010912	0.500000	7.02189

	antecedents	consequents	antecedent support	consequent support	support	confidence	lif
	WOODLAND CHARLOT...						

1397 rows × 10 columns

In [ ]:
```
rules.sort_values(by='support', ascending=False)
```

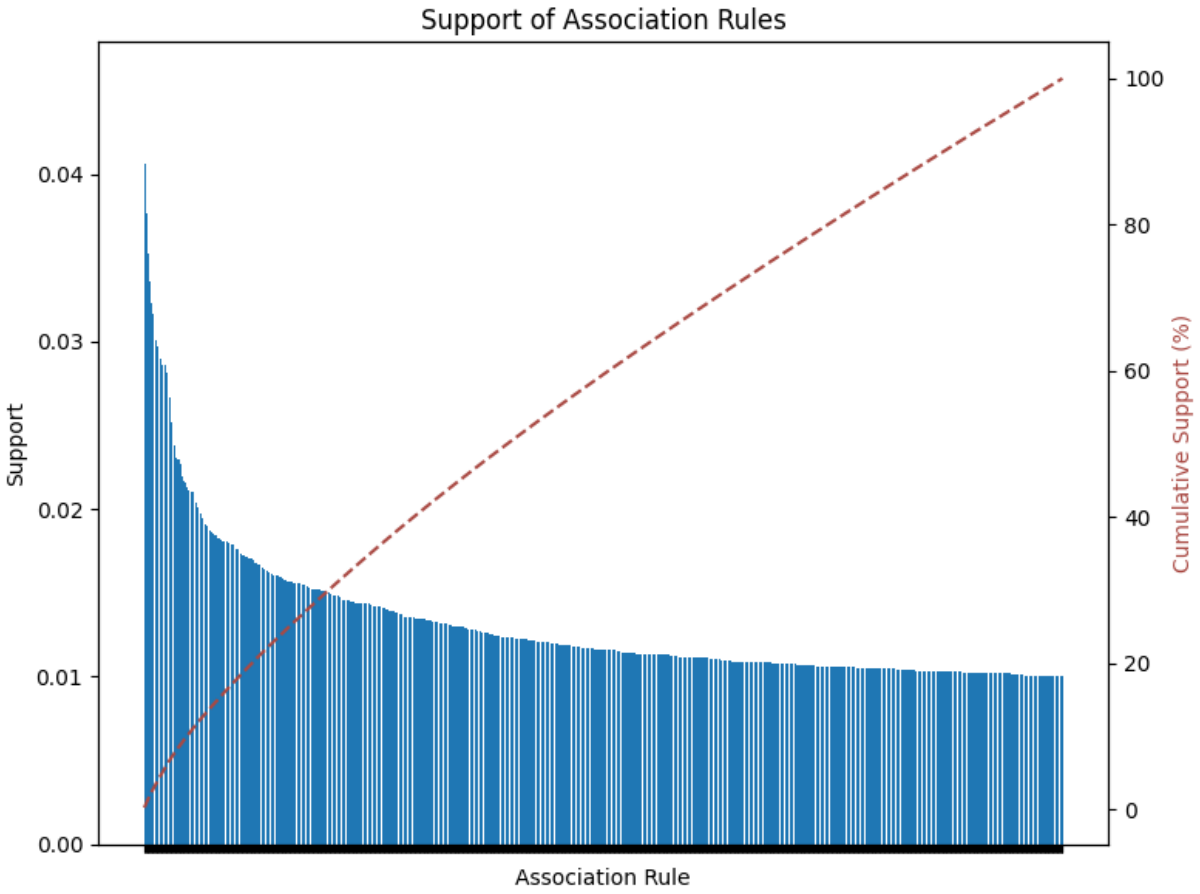
Out[ ]:

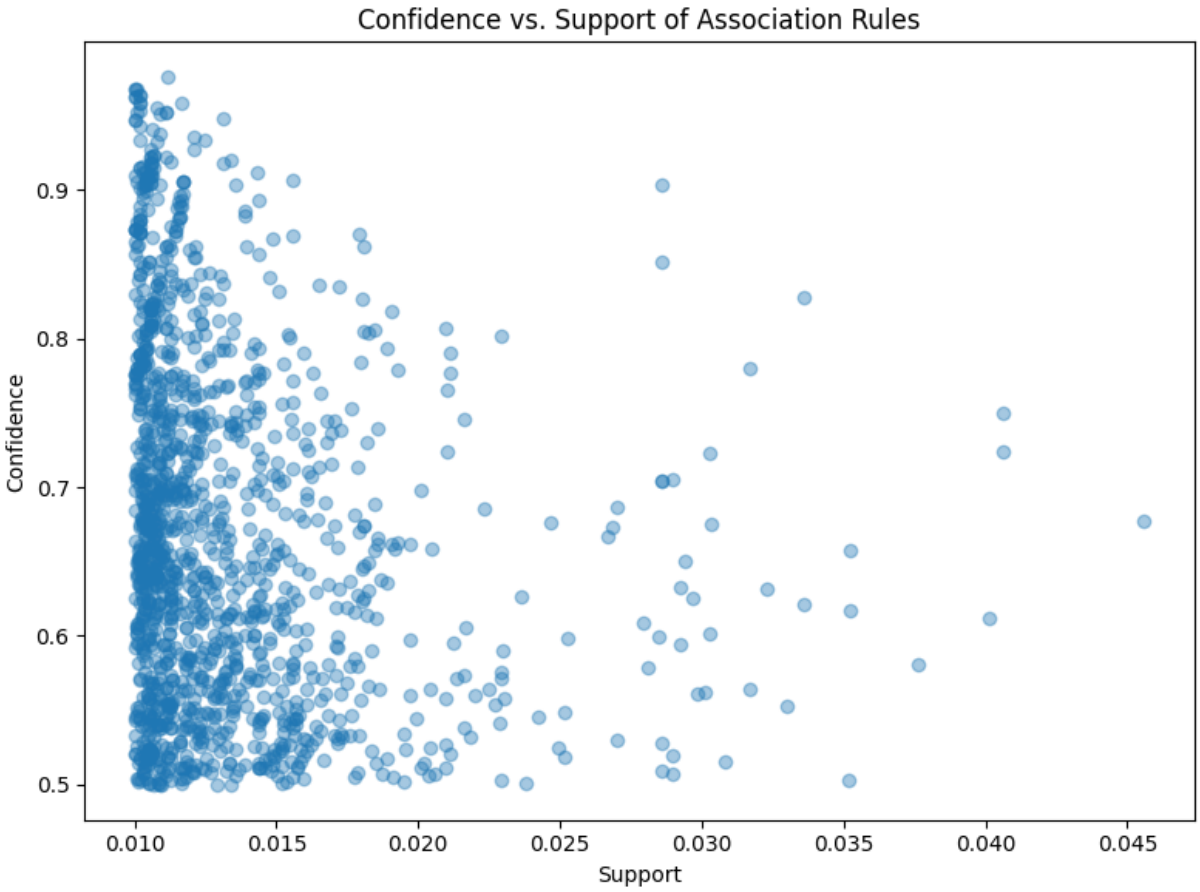
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
161	(JUMBO BAG PINK POLKADOT)	(JUMBO BAG RED RETROSPOT)	0.067309	0.113963	0.045596	0.677419	5.944214
105	(ROSES REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER)	0.056174	0.054170	0.040641	0.723489	13.355912
104	(GREEN REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.054170	0.056174	0.040641	0.750257	13.355912
174	(JUMBO STORAGE BAG SUKI)	(JUMBO BAG RED RETROSPOT)	0.065583	0.113963	0.040140	0.612054	5.370650
172	(JUMBO SHOPPER VINTAGE RED PAISLEY)	(JUMBO BAG RED RETROSPOT)	0.064859	0.113963	0.037635	0.580258	5.091639
...	...	...	...	...	...	...	...
608	(HERB MARKER ROSEMARY, HERB MARKER PARSLEY)	(HERB MARKER CHIVES)	0.011691	0.011469	0.010021	0.857143	74.737864
623	(HERB MARKER ROSEMARY)	(HERB MARKER CHIVES, HERB MARKER THYME)	0.013028	0.010411	0.010021	0.769231	73.887289
987	(LUNCH BAG PINK POLKADOT, LUNCH BAG APPLE DESIGN)	(LUNCH BAG SPACEBOY DESIGN)	0.019263	0.063857	0.010021	0.520231	8.146812
673	(LUNCH BAG RED RETROSPOT, JUMBO BAG BAROQUE B...	(JUMBO BAG RED RETROSPOT)	0.014364	0.113963	0.010021	0.697674	6.121948

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
431	(LUNCH BOX I LOVE LONDON, DOLLY GIRL LUNCH BOX)	(SPACEBOY LUNCH BOX)	0.014141	0.049215	0.010021	0.708661	14.399295

1397 rows × 10 columns

In [ ]:
```
# Sort rules by support in descending order
sorted_rules = rules.sort_values(by='support', ascending=False)

# Calculate cumulative support
cumulative_support = np.cumsum(sorted_rules['support'] / np.sum(sorted_rules['suppo

# Bar plot for Support
fig, ax1 = plt.subplots(figsize=(8, 6))
ax1.bar(range(len(sorted_rules)), sorted_rules['support'], align='center')
plt.xticks(range(len(sorted_rules)), ['' for _ in range(len(sorted_rules))])  # Rem
ax1.set_xlabel('Association Rule')
ax1.set_ylabel('Support')
ax1.set_title('Support of Association Rules')

# CDF plot for cumulative support
ax2 = ax1.twinx()
ax2.plot(range(len(sorted_rules)), cumulative_support, color='#AA4A44', linestyle='
ax2.set_ylabel('Cumulative Support (%)', c='#AA4A44')

plt.tight_layout()
plt.show()

# Scatter plot for Confidence vs. Support
plt.figure(figsize=(8, 6))
plt.scatter(rules['support'], rules['confidence'], alpha=0.4)
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Confidence vs. Support of Association Rules')
plt.tight_layout()
plt.show()
```


Support of Association Rules
Support
Cumulative Support (%)
Association Rule
0.00
0.01
0.02
0.03
0.04
0
20
40
60
80
100


Confidence vs. Support of Association Rules
Confidence
Support
0.5
0.6
0.7
0.8
0.9
0.010
0.015
0.020
0.025
0.030
0.035
0.040
0.045

These two visualizations explore the association rules: a bar plot for the support of association rules and a scatter plot for the confidence vs. support of association rules.

The bar plot represents the support values of the association rules. Each bar corresponds to a rule, and its height represents the support value, indicating how frequently the rule occurs in the dataset. The y-axis represents the support, while the x-axis does not display any labels, focusing solely on the visualization of support values.

The cumulative distribution function (CDF) plot showcases the cumulative support of the association rules as a percentage. It helps understand the distribution of support values across the rules in a cumulative manner. The red dashed line in the CDF plot connects the cumulative support values for each rule, providing insights into the accumulation of support as the rules progress.

The scatter plot displays the relationship between confidence and support for the association rules. Each point represents a rule, with the x-axis representing the support and the y-axis representing the confidence. The plot shows how the confidence varies with different levels of support, helping identify any patterns or trends between these two metrics.

These visualizations offer valuable insights into the support, confidence, and their relationships within the association rules, aiding in the interpretation and analysis of the rules' strength and significance.

Cross-Selling and Upselling

In [ ]:
```
# Filter association rules for cross-selling opportunities
cross_selling_rules = rules[(rules['antecedents'].apply(len) == 1) & (rules['conseq

# Sort rules based on confidence and support
cross_selling_rules = cross_selling_rules.sort_values(by=['confidence', 'support'],

# Select top cross-selling recommendations
top_cross_selling = cross_selling_rules.head(5)

# Filter association rules for upselling opportunities
upselling_rules = rules[(rules['antecedents'].apply(len) == 1) & (rules['consequent

# Sort rules based on confidence and support
upselling_rules = upselling_rules.sort_values(by=['confidence', 'support'], ascendi

# Select top upselling recommendations
top_upselling = upselling_rules.head(5)

# Display cross-selling recommendations
print("Cross-Selling Recommendations:")
for idx, row in top_cross_selling.iterrows():
    antecedent = list(row['antecedents'])[0]
    consequent = list(row['consequents'])[0]
```

```
    print(f"Customers who bought '{antecedent}' also bought '{consequent}'.")

# Display upselling recommendations
print("\nUpselling Recommendations:")
for idx, row in top_upselling.iterrows():
    antecedent = list(row['antecedents'])[0]
    consequents = list(row['consequents'])
    print(f"For customers who bought '{antecedent}', recommend the following upgrad
```

```
Cross-Selling Recommendations:
Customers who bought 'BEADED CRYSTAL HEART PINK ON STICK' also bought 'DOTCOM POSTAG
E'.
Customers who bought 'HERB MARKER THYME' also bought 'HERB MARKER ROSEMARY'.
Customers who bought 'HERB MARKER ROSEMARY' also bought 'HERB MARKER THYME'.
Customers who bought 'HERB MARKER CHIVES' also bought 'HERB MARKER PARSLEY'.
Customers who bought 'REGENCY TEA PLATE PINK' also bought 'REGENCY TEA PLATE GREEN'.

Upselling Recommendations:
For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HER
B MARKER PARSLEY, HERB MARKER THYME.
For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HER
B MARKER PARSLEY, HERB MARKER MINT.
For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HER
B MARKER ROSEMARY, HERB MARKER PARSLEY.
For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HER
B MARKER ROSEMARY, HERB MARKER THYME.
For customers who bought 'HERB MARKER THYME', recommend the following upgrades: HERB
MARKER ROSEMARY, HERB MARKER PARSLEY.
```

Upselling Recommendations

During the analysis of upselling opportunities, it was observed that multiple product recommendations were being made for the same item. To address this issue and provide more diverse recommendations, a modification was made to recommend only one product for each top item instead of recommending based on the top confidence values.

By implementing this change, we ensure that the upselling recommendations do not repeatedly suggest the same product to customers. This approach enhances the variety of product recommendations and increases the chances of cross-selling and upselling success.

The updated recommendation strategy focuses on identifying the top items and selecting a single recommended product for each of them. This adjustment aims to optimize the upselling strategy by suggesting different upgrades or add-on products to customers, resulting in a more compelling and varied range of recommendations.

In [ ]:
```
top_upselling = upselling_rules.sort_values(['confidence', 'support'], ascending=Fa
for idx, row in top_upselling.iterrows():
    antecedent = list(row['antecedents'])[0]
    consequents = list(row['consequents'])
    print(f"For customers who bought '{antecedent}', recommend the following upgrad
```

```
For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HER
B MARKER PARSLEY, HERB MARKER THYME.
For customers who bought 'HERB MARKER THYME', recommend the following upgrades: HERB
MARKER ROSEMARY, HERB MARKER PARSLEY.
For customers who bought 'HERB MARKER PARSLEY', recommend the following upgrades: HE
RB MARKER ROSEMARY, HERB MARKER THYME.
For customers who bought 'HERB MARKER ROSEMARY', recommend the following upgrades: H
ERB MARKER PARSLEY, HERB MARKER THYME.
For customers who bought 'REGENCY TEA PLATE PINK', recommend the following upgrades:
REGENCY TEA PLATE GREEN, REGENCY TEA PLATE ROSES.
```

Conclusion

In this project, we explored the concept of association rules using the Apriori algorithm and the mlxtend library in Python. Association rules analysis provides valuable insights into the relationships and patterns within a dataset, enabling businesses to uncover hidden associations between items and make informed decisions for various applications.

We started by preparing the data and filtering out infrequent items and irrelevant transactions. Then, we generated frequent itemsets and association rules based on predefined thresholds for support and confidence. These rules allowed us to identify significant associations between items and quantify their strength.

The generated association rules provided actionable insights for different business scenarios. We explored cross-selling opportunities by identifying products frequently purchased together. By leveraging these associations, businesses can implement effective cross-selling strategies, offering relevant add-on products or upgrades to customers, thereby increasing revenue.

Additionally, we examined upselling recommendations, focusing on identifying suitable product upgrades or higher-priced alternatives for customers. By considering only one product recommendation for each top item, we ensured diverse and relevant suggestions, avoiding repetitive recommendations and enhancing the upselling strategy.

Furthermore, we discussed the importance of interpreting the support, confidence, lift, leverage, and conviction metrics associated with association rules. These metrics provide quantitative measures of the strength, significance, and impact of the associations, enabling businesses to prioritize and optimize their decision-making processes.

Overall, association rules analysis offers valuable insights and practical applications across various domains, such as marketing, product recommendations, cross-selling strategies, and process optimization. By understanding the associations between items, businesses can make data-driven decisions, improve customer satisfaction, enhance marketing campaigns, and drive business growth.

It is important to note that the analysis and insights provided in this project are specific to the dataset and parameters used. The results can be further refined and customized based

on the specific requirements, domain knowledge, and business objectives.