

## NAAN MUDHALVAN PROJECT (IBM)

**IBM AI 101 ARTIFICIAL INTELLIGENCE-GROUP 1**

## Title: Market Basket Analysis

|  |  |
| --- | --- |
| **Team name**: unknown |  |
| **Team members:** Mohammed Hanifa M | 113321106052 |
| Naveen D | 113321106060 |
| Mohammed Tawfiq P | 113321106053 |
| Dharshan G | 113321106022 |
| Agathiyan G | 113321106002 |
| Jebastin D | 113321106039 |

**Problem Statement:**

Unveiling Customer Behaviour through Association Analysis: Utilize market basket analysis on the provided dataset to uncover hidden patterns and associations between products, aiming to understand customer purchasing behaviour and identify potential cross-selling opportunities for the retail business

Challenges:

Large dataset Market Basket Analysis can be computationally demanding when dealing with large datasets.

Efficient algorithms and scalable techniques are required to handle such data.

Sparse data problem When transactional data is sparse, meaning there are many items but few frequent item sets, Market Basket Analysis becomes more challenging.

Techniques like FP-growth algorithm can be utilized to address this problem. Interpretation of results Understanding and interpreting the generated association rules require domain knowledge and careful analysis.

False positives and irrelevant rules can mislead decision-making if not properly evaluated.

# **METHODS FOR MARKET BASKET ANALYSIS:**

### Association rule mining

Association rule mining is a popular technique used in Market Basket Analysis. It uncovers relationships and dependencies between items purchased together, enabling businesses to identify rules such as "If A is purchased, then B is likely to be purchased as well."

### Apriori algorithm

The Apriori algorithm is a widely used method for Market Basket Analysis. It scans the transactional data to discover frequent itemsets and generate association rules. It is efficient in handling large datasets and is highly interpretable.

### FP-growth algorithm

The FP-growth algorithm is an alternative approach to Association Rule Mining. It constructs a compact data structure called the FP-tree, which allows for faster pattern mining. It is particularly useful for handling sparse datasets.

**Data Mining**:

In this method we were concentrating on the customers and the retail shop owners they were bought the product depend upon the area need then the retail shop owners collect the data from the customers what they have bought daily basis in this daily needs are different but some company in the market selling some good products to the people in essential manner so depend upon this data the company which were selling good product increasing the product depend upon the area of the need this market basket analyzis is about we have done on the data mining.

**GOALS:**

1. Association Rule Discovery: Implement the Apriori algorithm to identify frequent item sets and generate meaningful association rules based on support, confidence, and lift.
2. Insightful Interpretation: Interpret the generated association rules to understand the relationships between products and their significance, translating them into actionable business strategies.
3. Business Recommendations: Provide clear and actionable recommendations to businesses based on the derived insights, empowering them to make informed decisions for marketing campaigns, inventory management, and customer experience enhancement.
4. Enhanced Sales and Customer Satisfaction: Ultimately, the goal is to enhance sales by optimizing product offerings and improving customer satisfaction through personalized recommendations and strategic business decisions.

**Design Thinking:**

In this project we have design the application model for this market basket analysis. This application is fully based on user friendly to the customers where they have bought the products on the nearby retail shop

# **The Customer Journey:**

#### Exploration

Customers embark on a journey to explore various products available in the retail store.

#### Selection

After exploring, customers carefully select the items they need for their basket.

#### Purchase

Customers proceed to the checkout and finalize their purchase.

# The Power of Data

#### Insights

By analyzing customer purchase data, we can gain valuable insights into their shopping habits.

#### Trends

Identifying trends helps us anticipate customer needs and tailor our offerings accordingly.

#### Personalization

With data-driven approaches, we can provide personalized recommendations to enhance the customer experience.

# **Streamlined Process**

#### Research

Gather data about customer preferences and shopping patterns.

#### Analyze

Apply advanced algorithms to identify purchasing patterns and trends.

#### Implement

Develop and integrate the market basket analysis application into retail shops.

# **Benefits for Retailers**

#### Optimized Inventory

Market basket analysis allows retailers to optimize their inventory based on customer demand.



#### Increased Sales

By offering personalized recommendations, retailers can increase sales and customer satisfaction.

#### Effective Marketing

With insights into customer preferences, retailers can develop targeted marketing strategies.



# **Finding Success through Data**

#### Customer-Centric Approach

By understanding customer needs, retailers can create personalized experiences and build brand loyalty.

#### Continuous Improvement

Regularly analyzing market basket data allows retailers to adapt and improve their offerings over time.

#### Staying Ahead

Using data-driven insights helps retailers stay ahead of competition and drive innovation in the industry



Market Analysis Model 1:

Market basket analysis is a technique used in retail and e-commerce to analyze customer purchase patterns and understand relationships between products that are frequently bought together. This analysis helps businesses optimize product placement, promotional strategies, and cross-selling efforts. One popular algorithm for market basket analysis is the Apriori algorithm, which generates association rules between items based on their co-occurrence in transactions.

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.max\_rows', None)

pd.set\_option('display.width', 500)

df = pd.read\_excel("/kaggle/input/market-basket-analysis/Assignment-1\_Data.xlsx")

def outlier\_thresholds(dataframe, variable):

quartile1 = dataframe[variable].quantile(0.01)

quartile3 = dataframe[variable].quantile(0.99)

interquantile\_range = quartile3 - quartile1

up\_limit = quartile3 + 1.5 \* interquantile\_range

low\_limit = quartile1 - 1.5 \* interquantile\_range

return low\_limit, up\_limit

def replace\_with\_thresholds(dataframe, variable):

low\_limit, up\_limit = outlier\_thresholds(dataframe, variable)

dataframe.loc[(dataframe[variable] < low\_limit), variable] = low\_limit

dataframe.loc[(dataframe[variable] > up\_limit), variable] = up\_limit

def retail\_data\_prep(dataframe):

dataframe = dataframe[dataframe["Quantity"] > 0]

dataframe = dataframe[dataframe["Price"] > 0]

replace\_with\_thresholds(dataframe, "Quantity")

replace\_with\_thresholds(dataframe, "Price")

return dataframe

df = retail\_data\_prep(df)

df.describe().T

Out[1]:

|  | count | mean | min | 25% | 50% | 75% | max | std |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Quantity | 519551.0 | 9.39742 | 1.0 | 1.0 | 3.0 | 10.0 | 248.5 | 21.281261 |
| Date | 519551 | 2011-07-04 16:03:31.051080704 | 2010-12-01 08:26:00 | 2011-03-28 10:52:00 | 2011-07-20 11:55:00 | 2011-10-19 15:08:00 | 2011-12-09 12:50:00 | NaN |
| Price | 519551.0 | 3.32647 | 0.001 | 1.25 | 2.08 | 4.13 | 41.94 | 3.87738 |
| CustomerID | 387985.0 | 15317.042994 | 12346.0 | 13950.0 | 15265.0 | 16837.0 | 18287.0 | 1721.813298 |

In [2]:

df\_fr = df[df['Country'] == "France"]

df\_fr.groupby(['BillNo', 'Itemname']).agg({"Quantity": "sum"}).unstack().fillna(0).iloc[0:5, 0:5]

Out[2]:

|  | Quantity | | | | |
| --- | --- | --- | --- | --- | --- |
| Itemname | 10 COLOUR SPACEBOY PEN | 12 COLOURED PARTY BALLOONS | 12 EGG HOUSE PAINTED WOOD | 12 MESSAGE CARDS WITH ENVELOPES | 12 PENCIL SMALL TUBE WOODLAND |
| BillNo |  |  |  |  |  |
| 536370 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 536852 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 536974 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 537065 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 537463 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

In [3]:

fr\_inv\_pro\_df=df\_fr.groupby(['BillNo', 'Itemname']). \

agg({"Quantity": "sum"}). \

unstack(). \

fillna(0). \

applymap(lambda x: 1 if x > 0 else 0)

frequent\_itemsets = apriori(fr\_inv\_pro\_df.astype("bool"),

min\_support=0.01,

use\_colnames=True)

frequent\_itemsets.sort\_values("support", ascending=False).head()

Out[3]:

|  | support | itemsets |
| --- | --- | --- |
| 330 | 0.765306 | ((Quantity, POSTAGE)) |
| 332 | 0.188776 | ((Quantity, RABBIT NIGHT LIGHT)) |
| 371 | 0.181122 | ((Quantity, RED TOADSTOOL LED NIGHT LIGHT)) |
| 320 | 0.170918 | ((Quantity, PLASTERS IN TIN WOODLAND ANIMALS)) |
| 315 | 0.168367 | ((Quantity, PLASTERS IN TIN CIRCUS PARADE)) |

Market Analysis Model 2:

Market basket analysis is a technique used in retail and e-commerce to discover associations and relationships between items that are frequently purchased together. The Apriori algorithm is a popular choice for this type of analysis. In this introduction, I'll provide you with Python code to perform basket analysis using the Apriori algorithm. We'll use the apyori library, which is a simple and lightweight library for Apriori

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import apriori, association\_rules

from pandas.plotting import parallel\_coordinates

import warnings

warnings.filterwarnings('ignore')

In [2]:

df = pd.read\_csv('../input/datasets-for-appiori/basket\_analysis.csv')

In [3]:

print(df)

Unnamed: 0 Apple Bread Butter Cheese Corn Dill Eggs Ice cream \

0 0 False True False False True True False True

1 1 False False False False False False False False

2 2 True False True False False True False True

3 3 False False True True False True False False

4 4 True True False False False False False False

.. ... ... ... ... ... ... ... ... ...

994 994 False True False False False False True False

995 995 True False False False True False False False

996 996 True False False False True True False False

997 997 False False True True True False True True

998 998 False False False False False False False False

Kidney Beans Milk Nutmeg Onion Sugar Unicorn Yogurt chocolate

0 False False False False True False True True

1 False True False False False False False False

2 False True False False False False True True

3 False True True True False False False False

4 False False False False False False False False

.. ... ... ... ... ... ... ... ...

994 False False False False False True False True

995 True True True False False False True False

996 False False False False True False False True

997 True False True False True False True True

998 False True False False False False False True

[999 rows x 17 columns]

In [4]:

df.drop(df.columns[0],axis=1,inplace=True)

In [5]:

print(df)

Apple Bread Butter Cheese Corn Dill Eggs Ice cream \

0 False True False False True True False True

1 False False False False False False False False

2 True False True False False True False True

3 False False True True False True False False

4 True True False False False False False False

.. ... ... ... ... ... ... ... ...

994 False True False False False False True False

995 True False False False True False False False

996 True False False False True True False False

997 False False True True True False True True

998 False False False False False False False False

Kidney Beans Milk Nutmeg Onion Sugar Unicorn Yogurt chocolate

0 False False False False True False True True

1 False True False False False False False False

2 False True False False False False True True

3 False True True True False False False False

4 False False False False False False False False

.. ... ... ... ... ... ... ... ...

994 False False False False False True False True

995 True True True False False False True False

996 False False False False True False False True

997 True False True False True False True True

998 False True False False False False False True

[999 rows x 16 columns]

In [6]:

df.shape

Out[6]:

(999, 16)

We have 999 basket for us to compute the recommendation for each item that sold in the store. There are 16 items that sold in the shop.

In [7]:

df.mean()

Out[7]:

Apple 0.383383

Bread 0.384384

Butter 0.420420

Cheese 0.404404

Corn 0.407407

Dill 0.398398

Eggs 0.384384

Ice cream 0.410410

Kidney Beans 0.408408

Milk 0.405405

Nutmeg 0.401401

Onion 0.403403

Sugar 0.409409

Unicorn 0.389389

Yogurt 0.420420

chocolate 0.421421

dtype: float64

The sale transaction or count for each unique item approximately for this sample. We will dive into and see whether there is any difference or correlation between the baskets. Since the dataframe is already tabulated one hot data frame, we will straight away and use the dataset to be analyzed with apriori

## Apriori Algorithm:

Little bit background introduction for Apriori Algorithm. The algorithm assumes that any subset of a frequent itemset must be frequent. Say in our cases, where {apple, unicorn, yoghurt} is frequent then {apple,yoghurt} is frequent. Whereas {apple,unicorn} is not frequent, then {apple,unicorn,yoghurt} is not frequent.

**SUPPORT** = A simple way to control complexity is to place a constraint that such rules must apply to some minimum percentage of the data  
**CONFIDENCE** = The probability that B occurs when A; it is p(B|A), which in association mining.  
**LIFT** = the co-occurrence of A and B is the probability that we actually see the two together, compared to the probability that we would see the two together if they were unrelated to (independent of) each other.  
**LEVERAGE** = alternative is to look at the difference between these quantities rather than their ratio.  
**CONVICTION** = measure to ascertain the direction of the rule. Unlike lift, conviction is sensitive to the rule direction.

Just Support and Confidence as a parameter might be misleading for items that are too common/ popular in the basket. It is more likely that popular items are part of the same basket just because they are popular rather than anything else.

We set the mininum support as 0.06, maximum number that being analysed in the basket is 3. We are doing first pruning and see what we get from the result

In [8]:

*# Compute frequent itemsets using the Apriori algorithm*

frequent\_itemsets = apriori(df,

min\_support = .006,

max\_len = 3,

use\_colnames = True)

In [9]:

frequent\_itemsets

Out[9]:

|  | support | itemsets |
| --- | --- | --- |
| 0 | 0.383383 | (Apple) |
| 1 | 0.384384 | (Bread) |
| 2 | 0.420420 | (Butter) |
| 3 | 0.404404 | (Cheese) |
| 4 | 0.407407 | (Corn) |
| ... | ... | ... |
| 691 | 0.098098 | (Onion, Yogurt, chocolate) |
| 692 | 0.087087 | (Yogurt, Sugar, Unicorn) |
| 693 | 0.090090 | (Sugar, chocolate, Unicorn) |
| 694 | 0.095095 | (Yogurt, Sugar, chocolate) |
| 695 | 0.086086 | (Yogurt, chocolate, Unicorn) |

In [10]:

*# Compute all association rules for frequent\_itemsets*

rules = association\_rules(frequent\_itemsets,

metric = 'support',

min\_threshold=0.1)

In [11]:

rules

Out[11]:

|  | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | (Apple) | (Bread) | 0.383383 | 0.384384 | 0.154154 | 0.402089 | 1.046059 | 0.006788 | 1.029610 |
| 1 | (Bread) | (Apple) | 0.384384 | 0.383383 | 0.154154 | 0.401042 | 1.046059 | 0.006788 | 1.029482 |
| 2 | (Butter) | (Apple) | 0.420420 | 0.383383 | 0.188188 | 0.447619 | 1.167549 | 0.027006 | 1.116289 |
| 3 | (Apple) | (Butter) | 0.383383 | 0.420420 | 0.188188 | 0.490862 | 1.167549 | 0.027006 | 1.138354 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | (Apple) | (Corn) | 0.383383 | 0.407407 | 0.186186 | 0.485640 | 1.192025 | 0.029993 | 1.152096 |
| 8 | (Apple) | (Dill) | 0.383383 | 0.398398 | 0.179179 | 0.467363 | 1.173104 | 0.026440 | 1.129478 |
| 15 | (Apple) | (Kidney Beans) | 0.383383 | 0.408408 | 0.176176 | 0.459530 | 1.125173 | 0.019599 | 1.094587 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 427 | (Kidney Beans, Yogurt) | (Nutmeg) | 0.194194 | 0.401401 | 0.101101 | 0.520619 | 1.297002 | 0.023151 | 1.248690 |
| 428 | (Nutmeg, Yogurt) | (Kidney Beans) | 0.192192 | 0.408408 | 0.101101 | 0.526042 | 1.288028 | 0.022608 | 1.248193 |
| 432 | (Yogurt, chocolate) | (Milk) | 0.198198 | 0.405405 | 0.104104 | 0.525253 | 1.295623 | 0.023753 | 1.252444 |
| 433 | (Yogurt, Milk) | (chocolate) | 0.190190 | 0.421421 | 0.104104 | 0.547368 | 1.298862 | 0.023954 | 1.278255 |
| 434 | (chocolate, Milk) | (Yogurt) | 0.211211 | 0.420420 | 0.104104 | 0.492891 | 1.172376 | 0.015307 | 1.142909 |

From the plot it seems like the butter can be used as cross-selling with other products, it also acts as something to be offered with antecedents that is low. Thus, the customers are more likely to buy them if the butter are offered with cheaper price if they buy the antecedents that sold less in a store

Market Analysis with apriori Model 3:

## | 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**

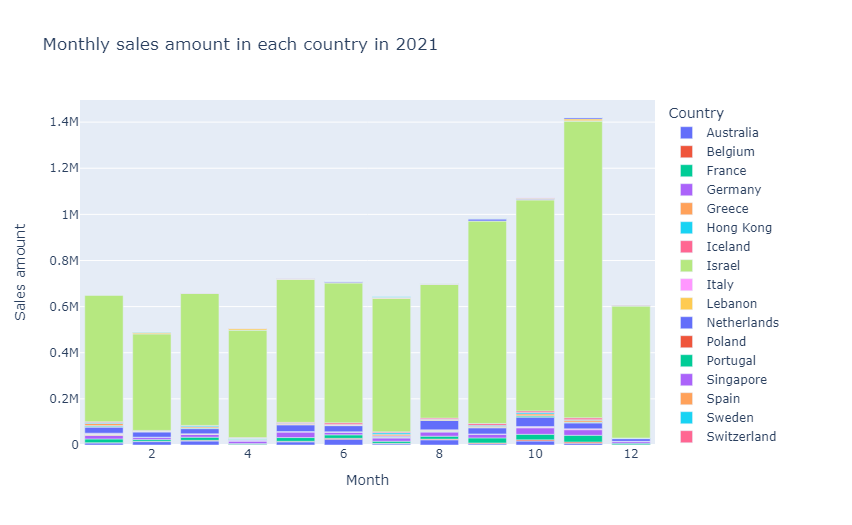
**Name: Total price, dtype: float64**

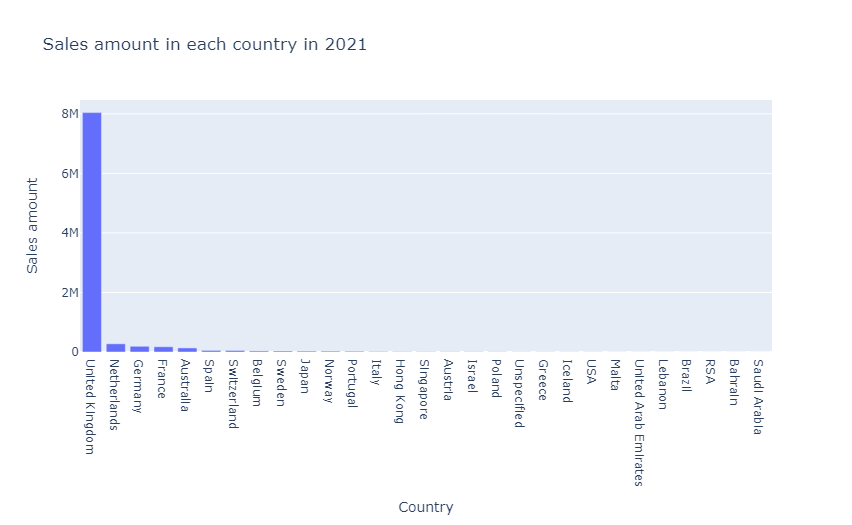
**It is appropriate to look at 12-month increments to implement data analytics properly, so I'll drop the data for 2020 Dec.**

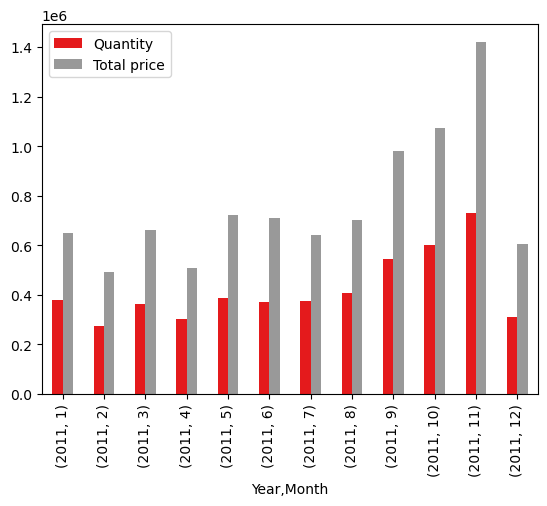
In [16]:

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

**OUTPUT:**

****

\



Market Analysis Trending Model 4:

kmeans = KMeans(n\_clusters = 3)

kmeans.fit(rfm\_scaled)

rfm\_data["Cluster\_No"] = (kmeans.labels\_ + 1)

In [18]:

rfm\_data.head()

Out[18]:

|  | Recency | Frequency | Monetary | Cluster\_No |
| --- | --- | --- | --- | --- |
| CustomerID |  |  |  |  |
| 12346.0 | 347 | 1 | 77183.60 | 3 |
| 12347.0 | 61 | 7 | 4310.00 | 2 |
| 12349.0 | 40 | 1 | 1757.55 | 2 |
| 12350.0 | 332 | 1 | 334.40 | 1 |
| 12352.0 | 94 | 8 | 2506.04 | 2 |

# **Analyzing of Clustering**

In [19]:

rfm\_data.groupby(["Cluster\_No"])[["Recency", "Frequency", "Monetary"]].mean()

Out[19]:

|  | Recency | Frequency | Monetary |
| --- | --- | --- | --- |
| Cluster\_No |  |  |  |
| 1 | 281.745299 | 1.545299 | 495.484189 |
| 2 | 68.634429 | 4.797872 | 1913.384218 |
| 3 | 48.760000 | 58.960000 | 81979.682000 |

Hmm. Our model determine **3 clusters** that

* **Cluster 1** --> Customers who haven't been here in a long time. We need to do some discount for them. We can still turn them back.
* **Cluster 2** --> Middle-level customers.
* **Cluster 3** --> Premium customers. We don't want to lose them. They spend a lot of money for us, and their recency is good.

In [20]:

plt.figure(figsize = (12,8))

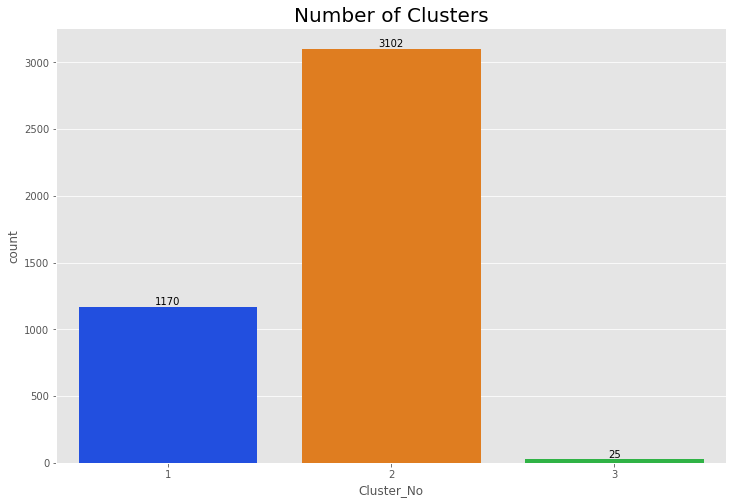
ax = sns.countplot(rfm\_data.Cluster\_No)

plt.title("Number of Clusters", fontsize = 20);

for bars **in** ax.containers:

ax.bar\_label(bars)

**OUTPUT:**

****

Market Analysis Model 5:

# BillNo Itemname **1) Import Libraries**[**¶**](https://www.kaggle.com/code/brcsnt/market-basket-analysis-wth-associationrulelearning#1)-Import-Libraries)

In [1]:

import pandas as pd

import numpy as np

from mlxtend.frequent\_patterns import apriori, association\_rules

import plotly.express as px

# **2) Data Pre-processing**

In [2]:

df\_ = pd.read\_csv("../input/market-basket-analysis/Assignment-1\_Data.csv", sep = ";")

df = df\_.copy()

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3444: DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or set low\_memory=False.

exec(code\_obj, self.user\_global\_ns, self.user\_ns)

In [3]:

df.head(10)

Out[3]:

|  | 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 |
| 5 | 536365 | SET 7 BABUSHKA NESTING BOXES | 2 | 01.12.2010 08:26 | 7,65 | 17850.0 | United Kingdom |
| 6 | 536365 | GLASS STAR FROSTED T-LIGHT HOLDER | 6 | 01.12.2010 08:26 | 4,25 | 17850.0 | United Kingdom |
| 7 | 536366 | HAND WARMER UNION JACK | 6 | 01.12.2010 08:28 | 1,85 | 17850.0 | United Kingdom |
| 8 | 536366 | HAND WARMER RED POLKA DOT | 6 | 01.12.2010 08:28 | 1,85 | 17850.0 | United Kingdom |
| 9 | 536367 | ASSORTED COLOUR BIRD ORNAMENT | 32 | 01.12.2010 08:34 | 1,69 | 13047.0 | United Kingdom |

In [4]:

def check\_df(dataframe, head=5):

print("##################### Shape #####################")

print(dataframe.shape)

print("##################### Types #####################")

print(dataframe.dtypes)

print("##################### Head #####################")

print(dataframe.head(head))

print("##################### Tail #####################")

print(dataframe.tail(head))

print("##################### NA #####################")

print(dataframe.isnull().sum())

In [5]:

check\_df(df)

##################### Shape #####################

(522064, 7)

##################### Types #####################

BillNo object

Itemname object

Quantity int64

Date object

Price object

CustomerID float64

Country object

dtype: object

##################### Head #####################

BillNo Itemname Quantity Date \

0 536365 WHITE HANGING HEART T-LIGHT HOLDER 6 01.12.2010 08:26

1 536365 WHITE METAL LANTERN 6 01.12.2010 08:26

2 536365 CREAM CUPID HEARTS COAT HANGER 8 01.12.2010 08:26

3 536365 KNITTED UNION FLAG HOT WATER BOTTLE 6 01.12.2010 08:26

4 536365 RED WOOLLY HOTTIE WHITE HEART. 6 01.12.2010 08:26

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

##################### Tail #####################

BillNo Itemname Quantity Date \

522059 581587 PACK OF 20 SPACEBOY NAPKINS 12 09.12.2011 12:50

522060 581587 CHILDREN'S APRON DOLLY GIRL 6 09.12.2011 12:50

522061 581587 CHILDRENS CUTLERY DOLLY GIRL 4 09.12.2011 12:50

522062 581587 CHILDRENS CUTLERY CIRCUS PARADE 4 09.12.2011 12:50

522063 581587 BAKING SET 9 PIECE RETROSPOT 3 09.12.2011 12:50

Price CustomerID Country

522059 0,85 12680.0 France

522060 2,1 12680.0 France

522061 4,15 12680.0 France

522062 4,15 12680.0 France

522063 4,95 12680.0 France

##################### NA #####################

BillNo 0

Itemname 1455

Quantity 0

Date 0

Price 0

CustomerID 134041

Country 0

dtype: int64

In [6]:

*# Drop na values*

df.dropna(inplace=True)

*# Quantity and Price should be greater than 0*

df = df[df["Quantity"] > 0]

*# We have to change the price column datatype as a numeric*

df ['Price'] = pd.to\_numeric(df['Price'], errors='coerce')

df = df[df["Price"] > 0]

In [7]:

check\_df(df)

##################### Shape #####################

(1537, 7)

##################### Types #####################

BillNo object

Itemname object

Quantity int64

Date object

Price float64

CustomerID float64

Country object

dtype: object

##################### Head #####################

BillNo Itemname Quantity Date \

45 536370 POSTAGE 3 01.12.2010 08:45

237 536392 RUSTIC SEVENTEEN DRAWER SIDEBOARD 1 01.12.2010 10:29

377 536403 POSTAGE 1 01.12.2010 11:27

1113 536527 POSTAGE 1 01.12.2010 13:04

4348 536779 Bank Charges 1 02.12.2010 15:08

Price CustomerID Country

45 18.0 12583.0 France

237 165.0 13705.0 United Kingdom

377 15.0 12791.0 Netherlands

1113 18.0 12662.0 Germany

4348 15.0 15823.0 United Kingdom

##################### Tail #####################

Quantity Date Price CustomerID \

521357 581493 POSTAGE 1 09.12.2011 10:10 15.0 12423.0

521375 581494 POSTAGE 2 09.12.2011 10:13 18.0 12518.0

521885 581570 POSTAGE 1 09.12.2011 11:59 18.0 12662.0

521922 581574 POSTAGE 2 09.12.2011 12:09 18.0 12526.0

521923 581578 POSTAGE 3 09.12.2011 12:16 18.0 12713.0

Country

521357 Belgium

521375 Germany

521885 Germany

521922 Germany

521923 Germany

##################### NA #####################

BillNo 0

Itemname 0

Quantity 0

Date 0

Price 0

CustomerID 0

Country 0

dtype: int64

# **3) Exploratory Data Analysis and Some Visualizations**

In [8]:

total\_sales = df

total\_sales["Total\_Price"] = total\_sales["Price"] \* total\_sales["Quantity"]

*#total\_sales.columns*

total\_sales\_per\_customer = total\_sales.groupby(["CustomerID", "Country"]).agg({"Total\_Price": "sum"})

total\_sales\_per\_customer.head(10)

Out[8]:

|  |  | Total\_Price |
| --- | --- | --- |
| CustomerID | Country |  |
| 12349.0 | Italy | 300.0 |
| 12350.0 | Norway | 40.0 |
| 12352.0 | Norway | 280.0 |
| 12356.0 | Portugal | 324.0 |
| 12357.0 | Switzerland | 25.0 |
| 12358.0 | Austria | 240.0 |
| 12360.0 | Austria | 360.0 |
| 12361.0 | Belgium | 15.0 |
| 12362.0 | Belgium | 489.0 |
| 12364.0 | Belgium | 105.0 |

## Top 10 Shoppers and Their Coutries

In [9]:

total\_sales\_per\_customer.reset\_index(inplace=True)

total\_sales\_per\_customer.sort\_values(by = "Total\_Price", ascending = False).head(10)

Out[9]:

|  | CustomerID | Country | Total\_Price |
| --- | --- | --- | --- |
| 577 | 17450.0 | United Kingdom | 10496.0 |
| 471 | 15581.0 | United Kingdom | 2750.0 |
| 67 | 12471.0 | Germany | 2400.0 |
| 412 | 14607.0 | United Kingdom | 2120.0 |
| 112 | 12540.0 | Spain | 1820.0 |
| 246 | 12748.0 | United Kingdom | 1788.0 |
| 460 | 15482.0 | United Kingdom | 1646.0 |
| 414 | 14646.0 | Netherlands | 1458.0 |
| 201 | 12681.0 | France | 1422.0 |
| 198 | 12678.0 | France | 1297.0 |

In [10]:

*# consider that for all time period*

data\_fig = total\_sales\_per\_customer.sort\_values(by = "Total\_Price", ascending = False).head(100)

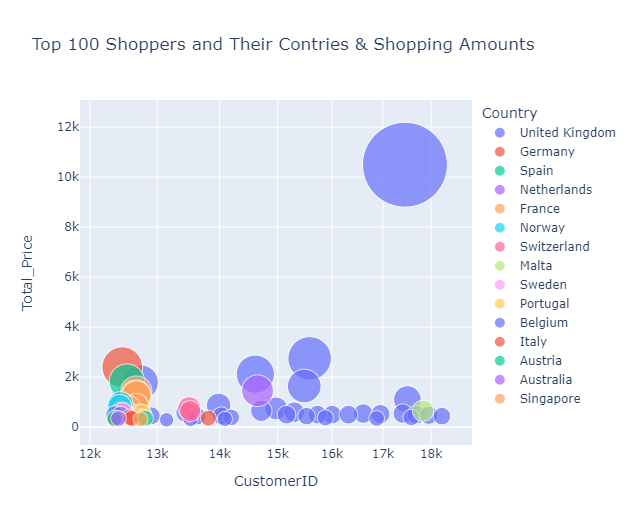
fig = px.scatter(data\_fig, x="CustomerID", y="Total\_Price",

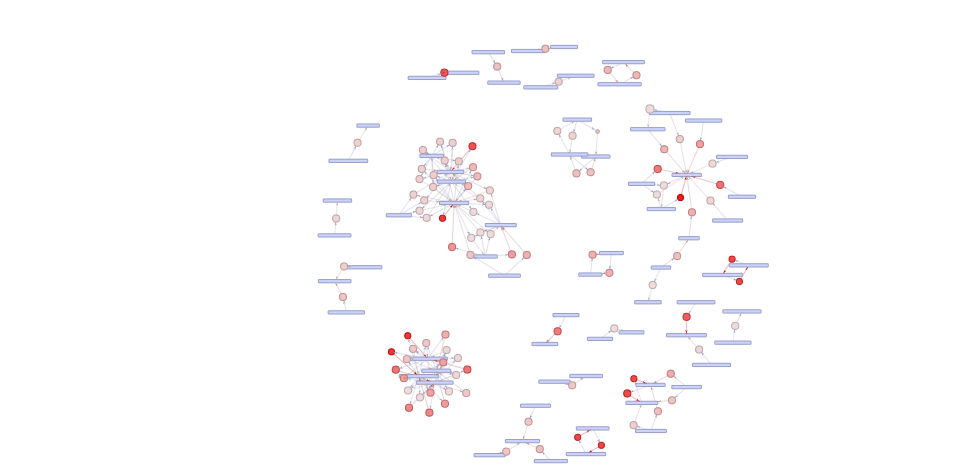
size="Total\_Price", color="Country",

hover\_name="Country", log\_x=True, size\_max=60, title="Top 100 Shoppers and Their Contries & Shopping Amounts")

fig.show()

**OUTPUT:**

****

****

# CONCLUSION:

Market Basket Analysis using AI is a valuable tool for businesses to gain a deeper understanding of customer behaviour and optimize sales strategies. By leveraging powerful algorithms and techniques, businesses can uncover hidden patterns and associations in customer purchase data, leading to improved decision-making, enhanced customer experience, and increased sales.