MARKET BASKET INSIGHTS

TEAM MEMBER

963521106027: DHANEESH.A

PHASE 4

PROJECT DEVELOPMENT PART 2

**TITLE:** MARKET BASKET INSIGHTS

**Abstract:**

Market basket insights (MBIs) is a data mining technique that identifies patterns and relationships between items purchased together in transactions. MBIs can be used to improve a variety of business processes, such as product placement, cross-selling, and targeted marketing.

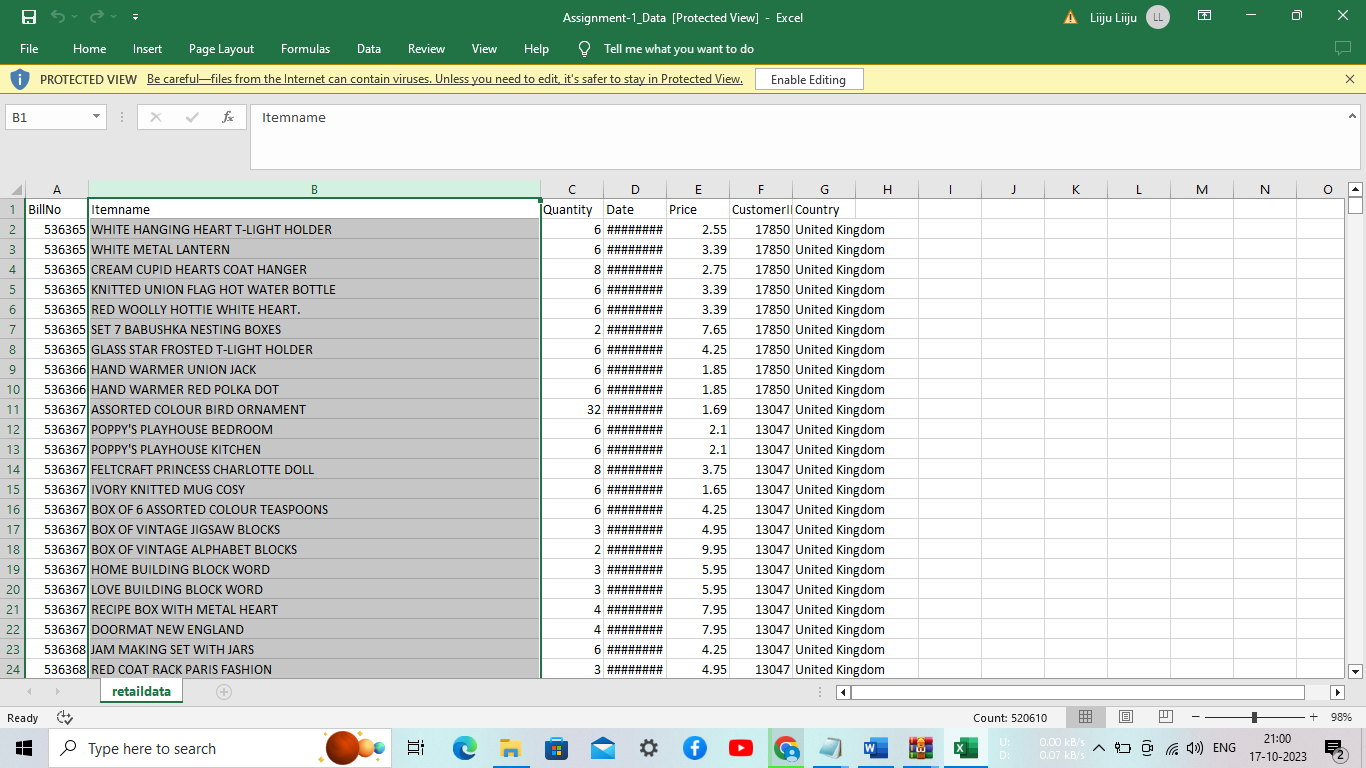


**Problem Definition:**

 Here we continue building the project by performing different activities like feature engineering, model training, evaluation etc

Dataset:

**Dataset Link:**[**https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis**](https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis)



**Feature Engineering:**

Feature engineering is the process of creating new features from existing data to improve the performance of a machine learning model. In the context of market basket analysis, feature engineering can be used to create new features that capture the relationships and patterns between items in a transactional dataset.

Here are some examples of feature engineering techniques for market basket analysis:

* Create binary features to indicate whether or not an item is present in a transaction. This is the most basic type of feature engineering for market basket analysis, but it can be very effective. For example, you could create a feature for each item in your dataset, with a value of 1 if the item is present in the transaction and a value of 0 if the item is not present.
* Create count features to indicate the number of times an item is purchased in a transaction. This can be useful for identifying items that are often purchased together. For example, you could create a feature for each pair of items in your dataset, with a value equal to the number of times the two items are purchased together in the same transaction.
* Create sequence features to capture the order in which items are purchased. This can be useful for identifying sequential patterns in the data. For example, you could create a feature for each pair of items in your dataset, with a value equal to the number of times the first item is purchased immediately before the second item.
* Create category features to group similar items together. This can be useful for reducing the dimensionality of the data and making it easier to identify patterns. For example, you could group all types of bread together into a single category, and then create a feature for each category, with a value equal to the number of items from that category that are purchased in the transaction.
* Create customer features to capture the characteristics of individual customers. This can be useful for identifying customer segments with similar purchasing behaviors. For example, you could create features for each customer, such as their age, gender, location, and purchase history.

The specific feature engineering techniques that you use will depend on the specific goals of your market basket analysis project. However, the examples above should give you a good starting point.

Here are some additional tips for feature engineering for market basket insights:

* Use domain knowledge to inform your feature engineering decisions. What are the business-specific relationships and patterns that you are interested in identifying? Once you understand the business goals of your project, you can start to think about what features would be most useful.
* Experiment with different feature engineering techniques and evaluate their performance on a holdout test set. There is no one-size-fits-all approach to feature engineering. The best features will vary depending on the specific dataset and the specific goals of the project.
* Use visualization tools to explore your data and identify potential features. Visualization tools can help you to identify patterns and relationships in your data that you might not have noticed otherwise.

By following these tips, you can create features that will help you to gain valuable insights from your market basket data.

**Sample code:**

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 [22]:

*# 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().plot()

*# 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().plot()

*# Adding legend and plot labels*

plt.legend(["2010", "2011"])

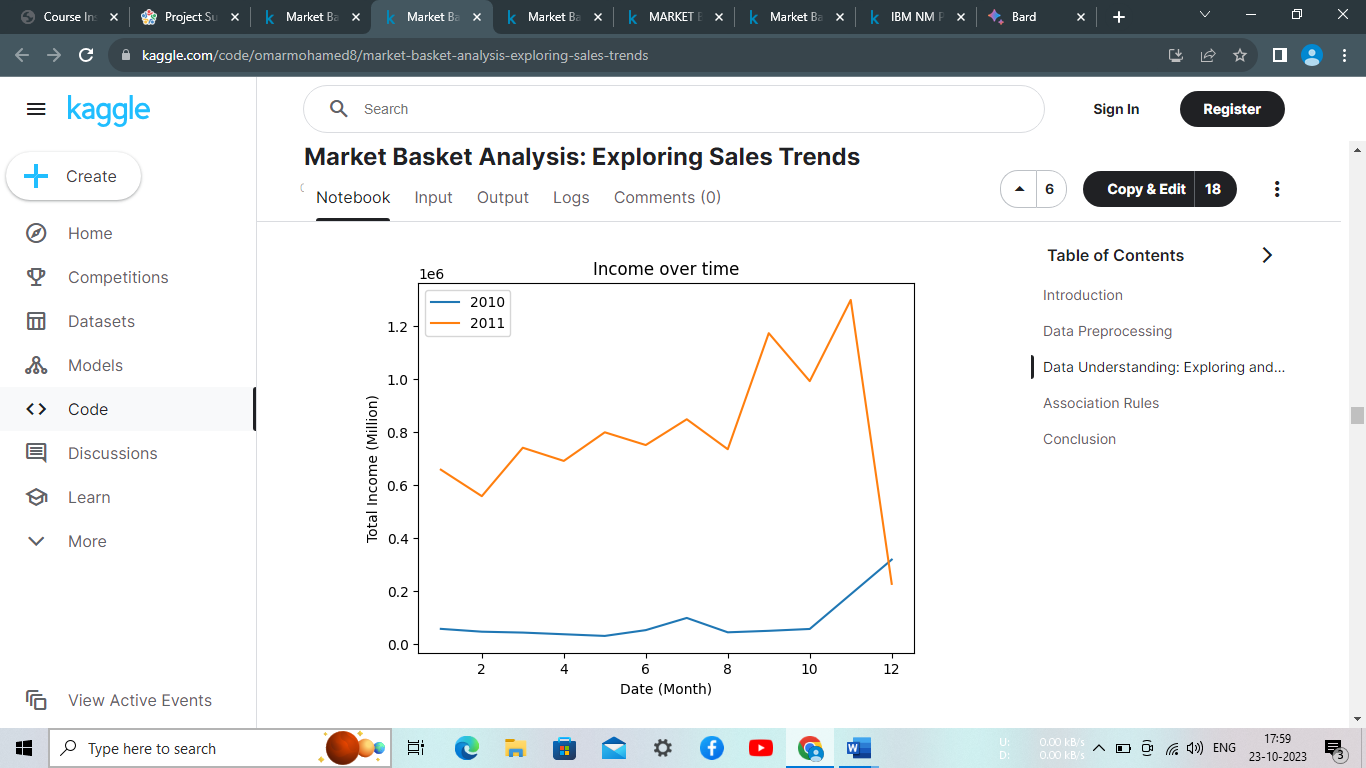
plt.title("Income over time")

plt.ylabel('Total Income (Million)')

plt.xlabel("Date (Month)")

Out[22]:

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 [23]:

df["Date"].max()

Out[23]:

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 [24]:

linkcode

*# Plotting the top 10 most sold products by quantity*

df.groupby('Itemname')['Quantity'].sum().sort\_values(ascending=False)[:10].plot(kind='barh', title='Number of Quantity Sold')

plt.ylabel('Item Name')

plt.xlim(20000, 82000)

plt.show()

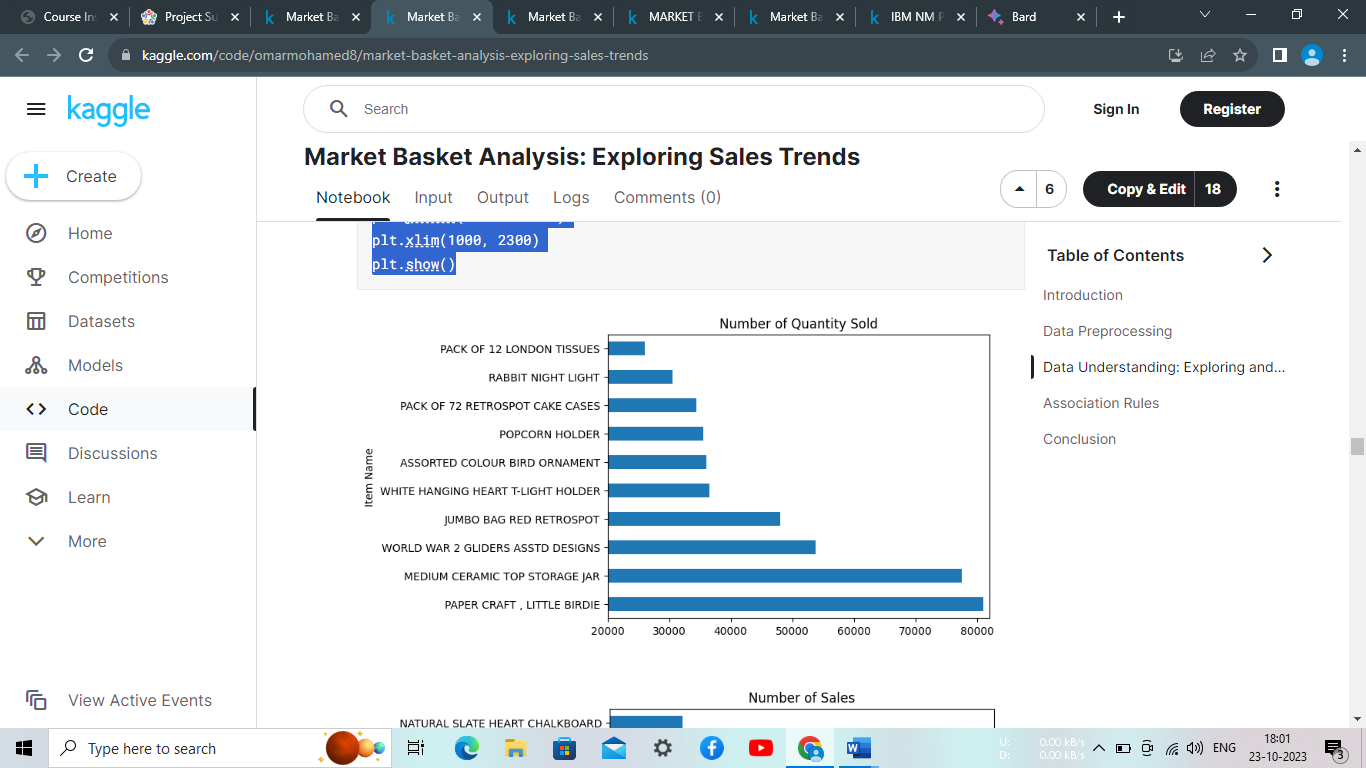
*# Plotting the top 10 most sold products by count*

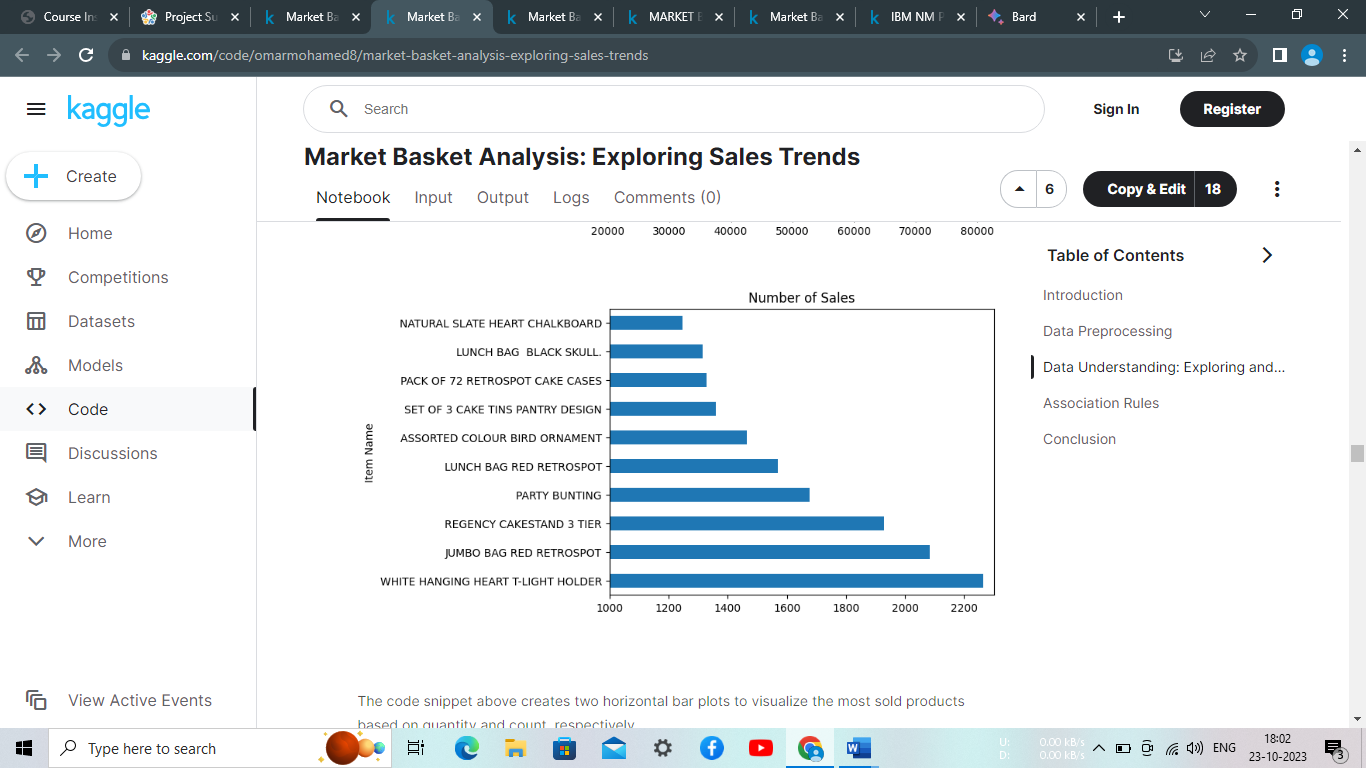
df['Itemname'].value\_counts(ascending=False)[:10].plot(kind='barh', title='Number of Sales')

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 [25]:

*# 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 [26]:

*# Create a pivot table using the filtered DataFrame*

pivot\_table = pd.pivot\_table(df2[['BillNo','Itemname']], index='BillNo', columns='Itemname', aggfunc=lambda x: True, fill\_value=False)

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 [27]:

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("**\n**Association 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[27]:

|  | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs\_metric |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | (60 CAKE CASES DOLLY GIRL DESIGN) | (PACK OF 72 RETROSPOT CAKE CASES) | 0.023160 | 0.071206 | 0.013028 | 0.562500 | 7.899629 | 0.011378 | 2.122958 | 0.894120 |
| 1 | (60 TEATIME FAIRY CAKE CASES) | (PACK OF 72 RETROSPOT CAKE CASES) | 0.044427 | 0.071206 | 0.024218 | 0.545113 | 7.655446 | 0.021054 | 2.041812 | 0.909794 |
| 2 | (ALARM CLOCK BAKELIKE CHOCOLATE) | (ALARM CLOCK BAKELIKE GREEN) | 0.023216 | 0.053558 | 0.015254 | 0.657074 | 12.268575 | 0.014011 | 2.759906 | 0.940321 |
| 3 | (ALARM CLOCK BAKELIKE CHOCOLATE) | (ALARM CLOCK BAKELIKE PINK) | 0.023216 | 0.042256 | 0.011691 | 0.503597 | 11.917802 | 0.010710 | 1.929369 | 0.937865 |
| 4 | (ALARM CLOCK BAKELIKE CHOCOLATE) | (ALARM CLOCK BAKELIKE RED) | 0.023216 | 0.057121 | 0.015811 | 0.681055 | 11.923112 | 0.014485 | 2.956246 | 0.937903 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1392 | (CHARLOTTE BAG SUKI DESIGN, STRAWBERRY CHARLOT... | (CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT... | 0.018483 | 0.021824 | 0.011302 | 0.611446 | 28.017319 | 0.010898 | 2.517477 | 0.982467 |
| 1393 | (CHARLOTTE BAG SUKI DESIGN, WOODLAND CHARLOTTE... | (CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL... | 0.018595 | 0.020989 | 0.011302 | 0.607784 | 28.957623 | 0.010911 | 2.496105 | 0.983760 |
| 1394 | (CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL... | (CHARLOTTE BAG SUKI DESIGN, WOODLAND CHARLOTTE... | 0.020989 | 0.018595 | 0.011302 | 0.538462 | 28.957623 | 0.010911 | 2.126378 | 0.986165 |
| 1395 | (CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT... | (CHARLOTTE BAG SUKI DESIGN, STRAWBERRY CHARLOT... | 0.021824 | 0.018483 | 0.011302 | 0.517857 | 28.017319 | 0.010898 | 2.035738 | 0.985822 |
| 1396 | (WOODLAND CHARLOTTE BAG, STRAWBERRY CHARLOTTE ... | (CHARLOTTE BAG PINK POLKADOT, CHARLOTTE BAG SU... | 0.022492 | 0.018261 | 0.011302 | 0.502475 | 27.516648 | 0.010891 | 1.973247 | 0.985832 |

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 [28]:

rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])

rules

Out[28]:

|  | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs\_metric |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 17 | (BEADED CRYSTAL HEART PINK ON STICK) | (DOTCOM POSTAGE) | 0.011469 | 0.039305 | 0.011190 | 0.975728 | 24.824404 | 0.010740 | 39.580626 | 0.970851 |
| 614 | (HERB MARKER CHIVES, HERB MARKER THYME) | (HERB MARKER PARSLEY) | 0.010411 | 0.012916 | 0.010077 | 0.967914 | 74.938272 | 0.009942 | 30.764113 | 0.997036 |
| 607 | (HERB MARKER CHIVES, HERB MARKER ROSEMARY) | (HERB MARKER PARSLEY) | 0.010355 | 0.012916 | 0.010021 | 0.967742 | 74.924917 | 0.009887 | 30.599599 | 0.996977 |
| 619 | (HERB MARKER CHIVES, HERB MARKER ROSEMARY) | (HERB MARKER THYME) | 0.010355 | 0.012916 | 0.010021 | 0.967742 | 74.924917 | 0.009887 | 30.599599 | 0.996977 |
| 1217 | (HERB MARKER BASIL, HERB MARKER ROSEMARY, HERB... | (HERB MARKER THYME) | 0.010578 | 0.012916 | 0.010188 | 0.963158 | 74.570009 | 0.010052 | 26.792276 | 0.997137 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 25 | (RED RETROSPOT CUP) | (BLUE POLKADOT CUP) | 0.021378 | 0.018038 | 0.010689 | 0.500000 | 27.719136 | 0.010304 | 1.963924 | 0.984981 |
| 1159 | (STRAWBERRY CHARLOTTE BAG, RED RETROSPOT CHARL... | (CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT... | 0.026834 | 0.021824 | 0.013417 | 0.500000 | 22.910714 | 0.012832 | 1.956352 | 0.982723 |
| 113 | (HAND WARMER RED LOVE HEART) | (HAND WARMER SCOTTY DOG DESIGN) | 0.021935 | 0.030286 | 0.010968 | 0.500000 | 16.509191 | 0.010303 | 1.939428 | 0.960496 |
| 147 | (LOVE HOT WATER BOTTLE) | (HOT WATER BOTTLE KEEP CALM) | 0.025832 | 0.042701 | 0.012916 | 0.500000 | 11.709257 | 0.011813 | 1.914597 | 0.938850 |
| 370 | (CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT... | (PACK OF 72 RETROSPOT CAKE CASES) | 0.021824 | 0.071206 | 0.010912 | 0.500000 | 7.021892 | 0.009358 | 1.857588 | 0.876722 |

1397 rows × 10 columns

In [29]:

rules.sort\_values(by='support', ascending=False)

Out[29]:

|  | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs\_metric |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 161 | (JUMBO BAG PINK POLKADOT) | (JUMBO BAG RED RETROSPOT) | 0.067309 | 0.113963 | 0.045596 | 0.677419 | 5.944214 | 0.037926 | 2.746715 | 0.891795 |
| 105 | (ROSES REGENCY TEACUP AND SAUCER) | (GREEN REGENCY TEACUP AND SAUCER) | 0.056174 | 0.054170 | 0.040641 | 0.723489 | 13.355912 | 0.037598 | 3.420583 | 0.980188 |
| 104 | (GREEN REGENCY TEACUP AND SAUCER) | (ROSES REGENCY TEACUP AND SAUCER) | 0.054170 | 0.056174 | 0.040641 | 0.750257 | 13.355912 | 0.037598 | 3.779187 | 0.978111 |
| 174 | (JUMBO STORAGE BAG SUKI) | (JUMBO BAG RED RETROSPOT) | 0.065583 | 0.113963 | 0.040140 | 0.612054 | 5.370650 | 0.032666 | 2.283921 | 0.870920 |
| 172 | (JUMBO SHOPPER VINTAGE RED PAISLEY) | (JUMBO BAG RED RETROSPOT) | 0.064859 | 0.113963 | 0.037635 | 0.580258 | 5.091639 | 0.030243 | 2.110907 | 0.859335 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 608 | (HERB MARKER ROSEMARY, HERB MARKER PARSLEY) | (HERB MARKER CHIVES) | 0.011691 | 0.011469 | 0.010021 | 0.857143 | 74.737864 | 0.009887 | 6.919719 | 0.998291 |
| 623 | (HERB MARKER ROSEMARY) | (HERB MARKER CHIVES, HERB MARKER THYME) | 0.013028 | 0.010411 | 0.010021 | 0.769231 | 73.887289 | 0.009886 | 4.288220 | 0.999487 |
| 987 | (LUNCH BAG PINK POLKADOT, LUNCH BAG APPLE DESIGN) | (LUNCH BAG SPACEBOY DESIGN) | 0.019263 | 0.063857 | 0.010021 | 0.520231 | 8.146812 | 0.008791 | 1.951238 | 0.894483 |
| 673 | (LUNCH BAG RED RETROSPOT, JUMBO BAG BAROQUE B... | (JUMBO BAG RED RETROSPOT) | 0.014364 | 0.113963 | 0.010021 | 0.697674 | 6.121948 | 0.008384 | 2.930738 | 0.848846 |
| 431 | (LUNCH BOX I LOVE LONDON, DOLLY GIRL LUNCH BOX) | (SPACEBOY LUNCH BOX) | 0.014141 | 0.049215 | 0.010021 | 0.708661 | 14.399295 | 0.009325 | 3.263505 | 0.943900 |

1397 rows × 10 columns

In [30]:

linkcode

*# 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['support']) \* 100)

*# 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))]) *# Remove x-axis labels*

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)

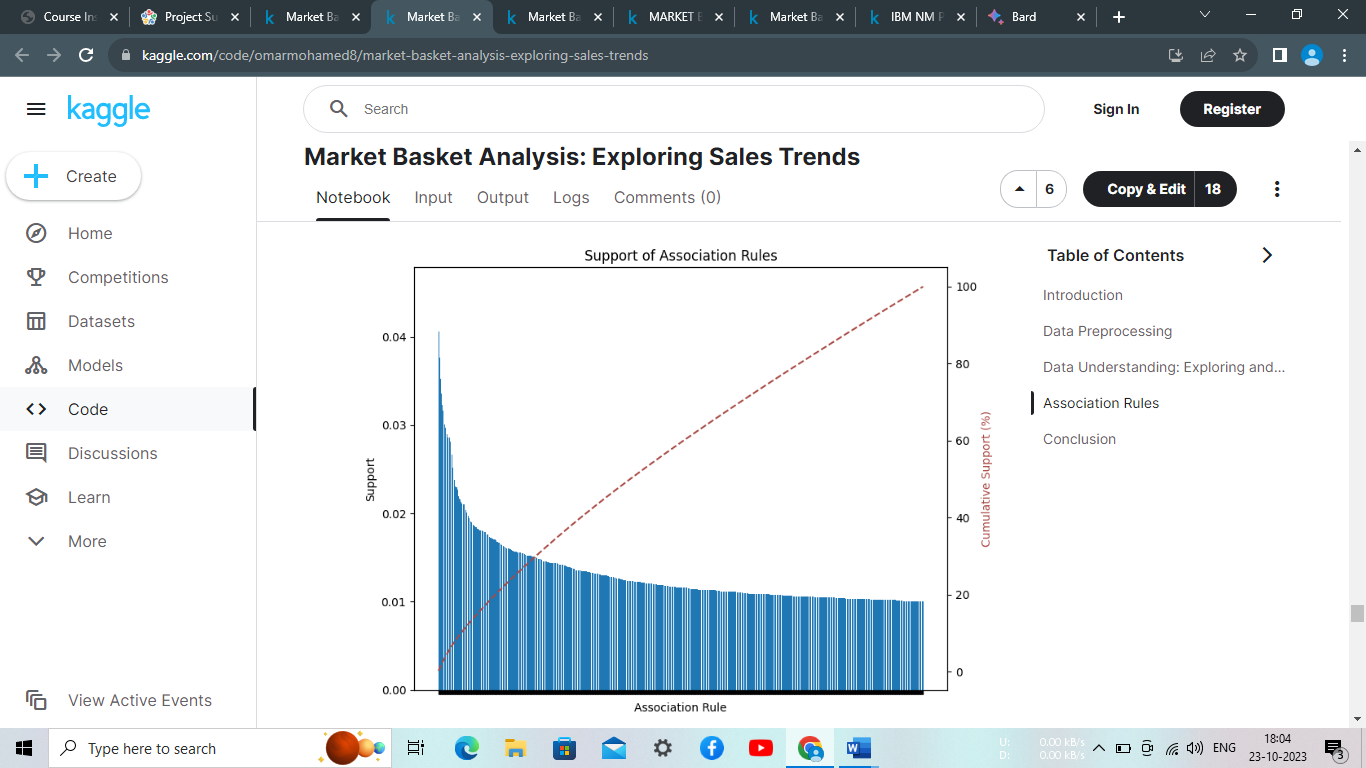
plt.xlabel('Support')

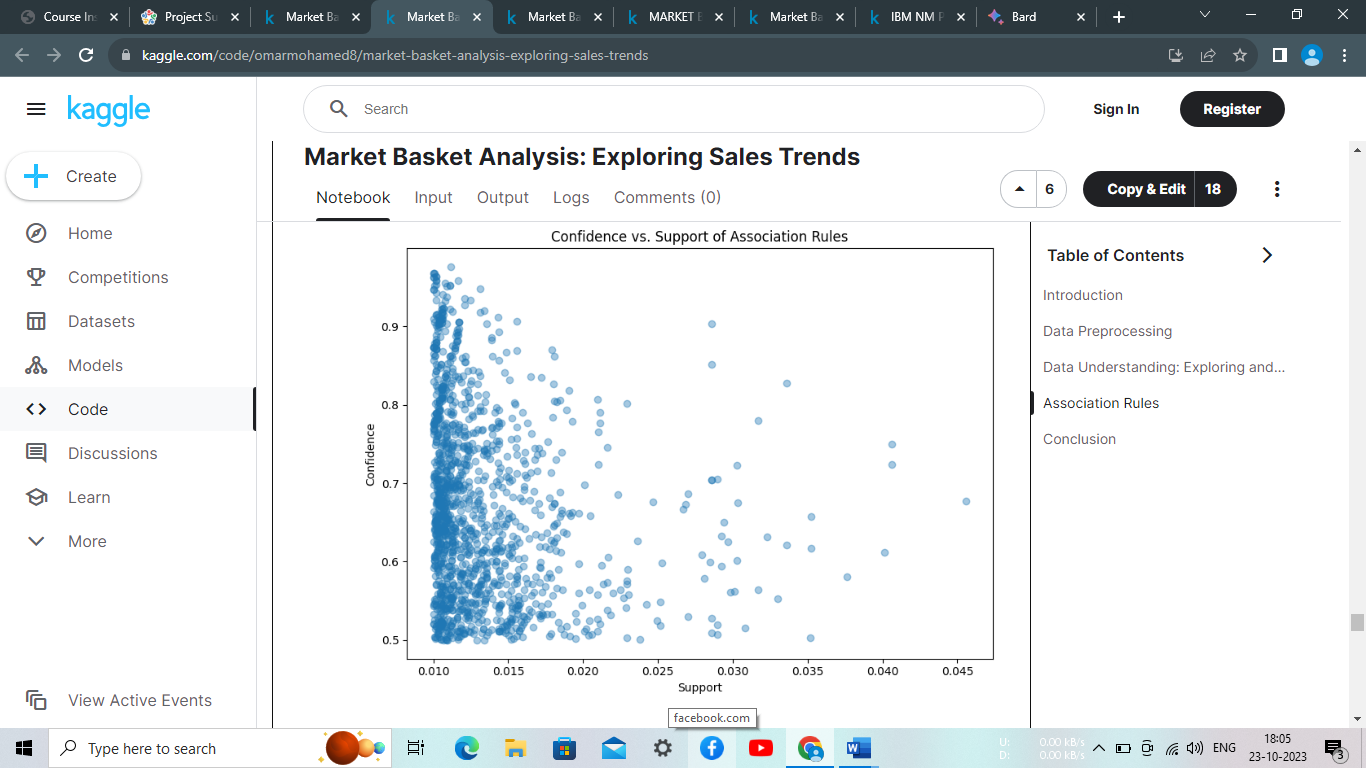
plt.ylabel('Confidence')

plt.title('Confidence vs. Support of Association Rules')

plt.tight\_layout()

plt.show()





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 [31]:

*# Filter association rules for cross-selling opportunities*

cross\_selling\_rules = rules[(rules['antecedents'].apply(len) == 1) & (rules['consequents'].apply(len) == 1)]

*# Sort rules based on confidence and support*

cross\_selling\_rules = cross\_selling\_rules.sort\_values(by=['confidence', 'support'], ascending=False)

*# 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['consequents'].apply(len) > 1)]

*# Sort rules based on confidence and support*

upselling\_rules = upselling\_rules.sort\_values(by=['confidence', 'support'], ascending=False)

*# 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("**\n**Upselling 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 upgrades: **{**', '.join(consequents)**}**.")

Cross-Selling Recommendations:

Customers who bought 'BEADED CRYSTAL HEART PINK ON STICK' also bought 'DOTCOM POSTAGE'.

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: HERB MARKER PARSLEY, HERB MARKER THYME.

For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HERB MARKER PARSLEY, HERB MARKER MINT.

For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HERB MARKER ROSEMARY, HERB MARKER PARSLEY.

For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HERB 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 [32]:

top\_upselling = upselling\_rules.sort\_values(['confidence', 'support'], ascending=False).drop\_duplicates('antecedents')[:5]

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 upgrades: **{**', '.join(consequents)**}**.")

For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HERB 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: HERB MARKER ROSEMARY, HERB MARKER THYME.

For customers who bought 'HERB MARKER ROSEMARY', recommend the following upgrades: HERB 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 mlx tend 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.

**Model Training:**

Model training for market basket insights is the process of using a machine learning algorithm to learn the relationships and patterns between items in a transactional dataset. Once the model is trained, it can be used to generate insights into customer behaviour, identify product associations, and optimize pricing and promotions.

There are a number of different machine learning algorithms that can be used for market basket analysis. Some of the most popular algorithms include:

* **Apriori algorithm**: The Apriori algorithm is a classic association rule learning algorithm. It works by identifying frequent item sets and then generating association rules from those frequent item sets.
* **FP-growth algorithm:** The FP-growth algorithm is another popular association rule learning algorithm. It is more efficient than the Apriori algorithm for large datasets.
* **Markov chain models**: Markov chain models can be used to model the sequential relationships between items in a transactional dataset. This can be useful for identifying sequential patterns in the data, such as which items are often purchased together in sequence.
* Deep learning models: Deep learning models can also be used for market basket analysis. However, deep learning models are typically more complex and require more data to train than other types of machine learning models.

The specific machine learning algorithm that you choose will depend on the specific goals of your market basket analysis project and the size and complexity of your dataset.

Here are some general steps for model training for market basket insights:

1. **Prepare your data.** This includes cleaning the data, removing outliers, and encoding the data in a format that is compatible with your chosen machine learning algorithm.
2. **Choose a machine learning algorithm**. As mentioned above, there are a number of different machine learning algorithms that can be used for market basket analysis. Choose an algorithm that is appropriate for the specific goals of your project and the size and complexity of your dataset.
3. **Train the model**. This involves feeding the prepared data to the machine learning algorithm and allowing it to learn the relationships and patterns in the data.
4. **Evaluate the model**. Once the model is trained, you need to evaluate its performance on a holdout test set. This will help you to identify any potential problems with the model and make necessary adjustments.
5. **Deploy the model**. Once you are satisfied with the performance of the model, you can deploy it to production. This may involve integrating the model into a business application or making it available as a web service.

Once the model is deployed, you can use it to generate insights into customer behaviour, identify product associations, and optimize pricing and promotions. For example, you could use the model to identify products that are often purchased together and then bundle those products together in promotions. Or, you could use the model to identify products that are frequently purchased by certain customer segments and then target those customers with personalized marketing campaigns.

Here are some additional tips for model training for market basket insights:

* **Use a large and diverse dataset**. The more data you have to train your model, the better it will perform. Make sure to use a dataset that represents the full range of customer behaviour and product associations.
* **Use a variety of feature engineering techniques**. Feature engineering can help you to improve the performance of your model by creating new features that capture the relationships and patterns in your data.
* **Tune the hyperparameters of your model**. Hyperparameters are parameters that control the behaviour of the machine learning algorithm. Tuning the hyperparameters of your model can help you to improve its performance on the training data and the test data.
* **Monitor the performance of your model over time**. Customer behaviour and product associations can change over time, so it is important to monitor the performance of your model over time and retrain it as needed.

By following these tips, you can train a model that will help you to gain valuable insights from your market basket data.

**Sample code:**

# **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 #####################

BillNo Itemname 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]:

linkcode

*# consider that for all time period*

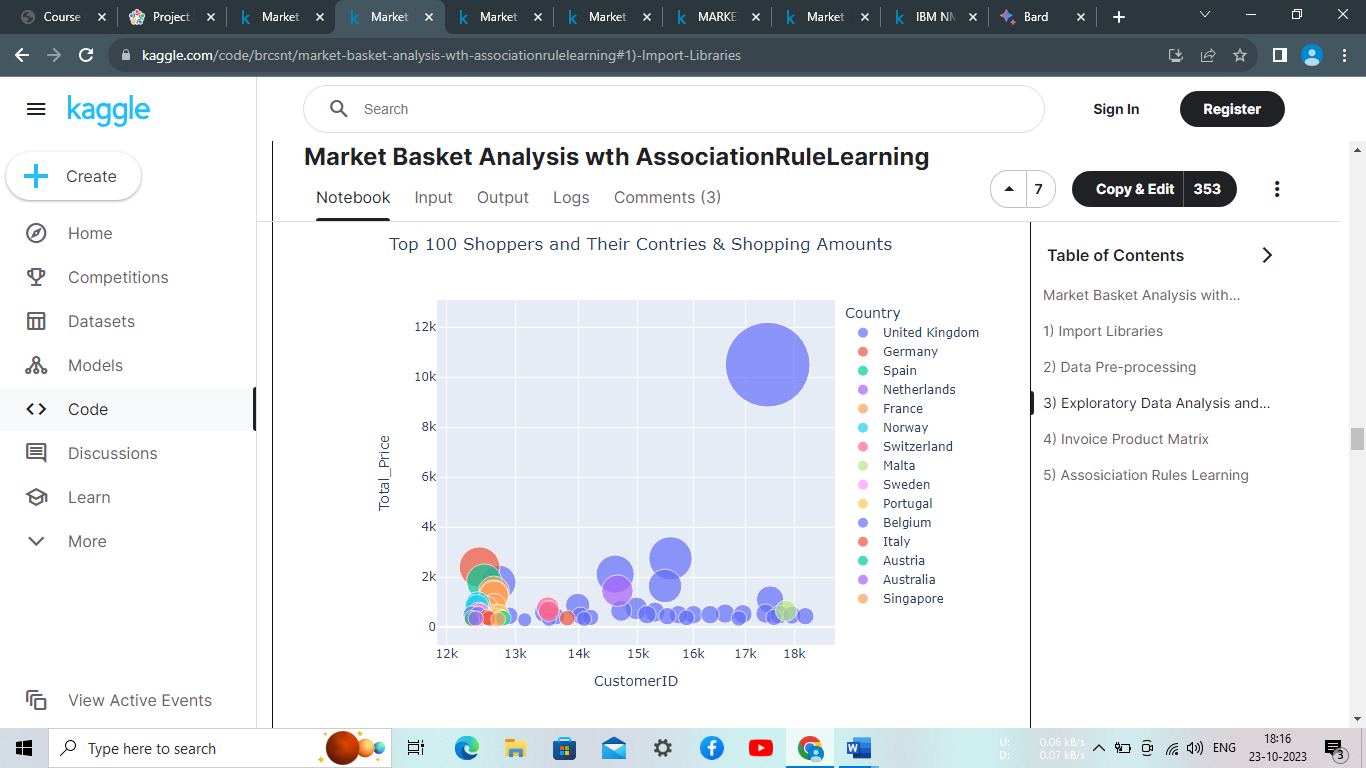
data\_fig = total\_sales\_per\_customer.sort\_values(by = "Total\_Price", ascending = False).head(100)

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()



*# consider that for all time period*

*#total\_sales\_per\_customer.head(20)*

total\_sales\_per\_customer.groupby(["Country"]).agg({"Total\_Price":"sum"}).reset\_index().sort\_values(by="Total\_Price", ascending=False )

Out[11]:

|  | Country | Total\_Price |
| --- | --- | --- |
| 17 | United Kingdom | 53159.0 |
| 4 | Germany | 21155.0 |
| 3 | France | 15713.0 |
| 14 | Spain | 5917.0 |
| 2 | Belgium | 4269.0 |
| 16 | Switzerland | 4027.0 |
| 10 | Norway | 2996.0 |
| 12 | Portugal | 2508.0 |
| 9 | Netherlands | 1947.0 |
| 7 | Italy | 1663.0 |
| 15 | Sweden | 1539.0 |
| 1 | Austria | 1456.0 |
| 8 | Malta | 655.0 |
| 11 | Poland | 360.0 |
| 0 | Australia | 350.0 |
| 5 | Greece | 335.0 |
| 13 | Singapore | 315.0 |
| 6 | Israel | 255.0 |

In [12]:

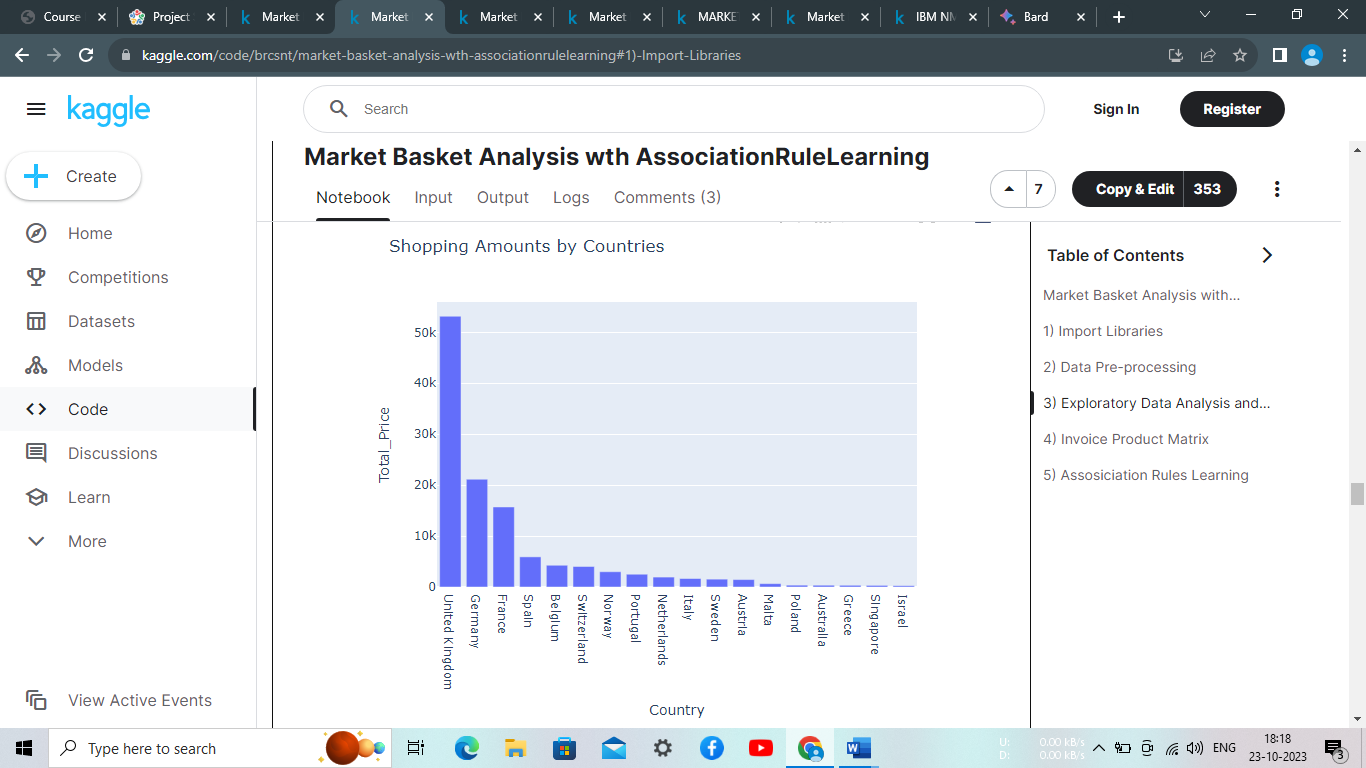
linkcode

*# consider that for all time period*

data = total\_sales\_per\_customer.groupby(["Country"]).agg({"Total\_Price":"sum"}).reset\_index().sort\_values(by="Total\_Price", ascending=False )

fig = px.bar(data, x='Country', y='Total\_Price' , title = "Shopping Amounts by Countries")

fig.show()



# **4) Invoice Product Matrix**

In [13]:

*#df\_united\_kingdom = df.loc[df["Country"]=="United Kingdom"]*

df\_invoice\_product\_matrix = df.groupby(['BillNo', 'Itemname']). \

agg({"Quantity": "sum"}).unstack().fillna(0). \

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

df\_invoice\_product\_matrix.head(10)

Out[13]:

|  | Quantity | | | | | | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Itemname | BEADED CHANDELIER T-LIGHT HOLDER | BILI NUT AND WOOD NECKLACE | BLING KEY RING STAND | BOTANICAL GARDENS WALL CLOCK | BREAD BIN DINER STYLE PINK | BREAD BIN DINER STYLE RED | BROWN CHECK CAT DOORSTOP | BROWN KUKUI COCONUT SEED NECKLACE | Bank Charges | CAKE STAND VICTORIAN FILIGREE MED | ... | SMALL WHITE RETROSPOT MUG IN BOX | SPOTTED WHITE NATURAL SEED NECKLACE | TWO DOOR CURIO CABINET | UTILTY CABINET WITH HOOKS | VANILLA SCENT CANDLE JEWELLED BOX | VICTORIAN SEWING BOX SMALL | VINTAGE BLUE KITCHEN CABINET | VINTAGE RED KITCHEN CABINET | VINTAGE WOODEN BAR STOOL | WOODEN ADVENT CALENDAR CREAM |
| BillNo |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 536370 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536392 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536403 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536527 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536779 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536835 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 536840 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536852 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536858 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536861 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

10 rows × 73 columns

# **5) Association Rules Learning**

In [14]:

frequent\_itemsets = apriori(df\_invoice\_product\_matrix, min\_support=0.001, use\_colnames=True)

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

Out[14]:

|  | support | itemsets |
| --- | --- | --- |
| 30 | 0.715076 | ((Quantity, POSTAGE)) |
| 27 | 0.054633 | ((Quantity, Next Day Carriage)) |
| 3 | 0.043568 | ((Quantity, BOTANICAL GARDENS WALL CLOCK)) |
| 26 | 0.031120 | ((Quantity, Manual)) |
| 23 | 0.027663 | ((Quantity, LOVE SEAT ANTIQUE WHITE METAL)) |
| ... | ... | ... |
| 573 | 0.001383 | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... |
| 574 | 0.001383 | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... |
| 575 | 0.001383 | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... |
| 576 | 0.001383 | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... |
| 1084 | 0.001383 | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... |

1085 rows × 2 columns

In [15]:

rules = association\_rules(frequent\_itemsets, metric="support", min\_threshold=0.001)

rules.sort\_values("support", ascending=False).head(10)

Out[15]:

|  | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 123 | ((Quantity, VINTAGE BLUE KITCHEN CABINET)) | ((Quantity, VINTAGE RED KITCHEN CABINET)) | 0.012448 | 0.024896 | 0.006916 | 0.555556 | 22.314815 | 0.006606 | 2.193983 |
| 122 | ((Quantity, VINTAGE RED KITCHEN CABINET)) | ((Quantity, VINTAGE BLUE KITCHEN CABINET)) | 0.024896 | 0.012448 | 0.006916 | 0.277778 | 22.314815 | 0.006606 | 1.367380 |
| 17 | ((Quantity, SPOTTED WHITE NATURAL SEED NECKLACE)) | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)) | 0.003458 | 0.004149 | 0.002766 | 0.800000 | 192.800000 | 0.002752 | 4.979253 |
| 14 | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)) | ((Quantity, RED KUKUI COCONUT SEED NECKLACE)) | 0.004149 | 0.007607 | 0.002766 | 0.666667 | 87.636364 | 0.002735 | 2.977178 |
| 137 | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)... | ((Quantity, RED KUKUI COCONUT SEED NECKLACE)) | 0.002766 | 0.007607 | 0.002766 | 1.000000 | 131.454545 | 0.002745 | inf |
| 138 | ((Quantity, RED KUKUI COCONUT SEED NECKLACE), ... | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)) | 0.002766 | 0.004149 | 0.002766 | 1.000000 | 241.000000 | 0.002755 | inf |
| 139 | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)) | ((Quantity, RED KUKUI COCONUT SEED NECKLACE), ... | 0.004149 | 0.002766 | 0.002766 | 0.666667 | 241.000000 | 0.002755 | 2.991701 |
| 140 | ((Quantity, RED KUKUI COCONUT SEED NECKLACE)) | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)... | 0.007607 | 0.002766 | 0.002766 | 0.363636 | 131.454545 | 0.002745 | 1.567082 |
| 141 | ((Quantity, BILI NUT AND WOOD NECKLACE)) | ((Quantity, BROWN KUKUI COCONUT SEED NECKLACE)... | 0.002766 | 0.002766 | 0.002766 | 1.000000 | 361.500000 | 0.002759 | inf |
| 119 | ((Quantity, SPOTTED WHITE NATURAL SEED NECKLACE)) | ((Quantity, RED KUKUI COCONUT SEED NECKLACE)) | 0.003458 | 0.007607 | 0.002766 | 0.800000 | 105.163636 | 0.002740 | 4.961964 |

In [16]:

sorted\_rules = rules.sort\_values("lift", ascending=False)

rules.sort\_values("lift", ascending=False).head(10)

Out[16]:

|  | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 113 | ((Quantity, MARIE ANTOINETTE TRINKET BOX GOLD)) | ((Quantity, MARIE ANTOINETTE TRINKET BOX SILVER)) | 0.001383 | 0.001383 | 0.001383 | 1.000000 | 723.0 | 0.001381 | inf |
| 114 | ((Quantity, VANILLA SCENT CANDLE JEWELLED BOX)) | ((Quantity, OCEAN SCENT CANDLE IN JEWELLED BOX)) | 0.001383 | 0.001383 | 0.001383 | 1.000000 | 723.0 | 0.001381 | inf |
| 112 | ((Quantity, MARIE ANTOINETTE TRINKET BOX SILVER)) | ((Quantity, MARIE ANTOINETTE TRINKET BOX GOLD)) | 0.001383 | 0.001383 | 0.001383 | 1.000000 | 723.0 | 0.001381 | inf |
| 115 | ((Quantity, OCEAN SCENT CANDLE IN JEWELLED BOX)) | ((Quantity, VANILLA SCENT CANDLE JEWELLED BOX)) | 0.001383 | 0.001383 | 0.001383 | 1.000000 | 723.0 | 0.001381 | inf |
| 38147 | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... | 0.002075 | 0.001383 | 0.001383 | 0.666667 | 482.0 | 0.001380 | 2.995851 |
| 38078 | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... | ((Quantity, DOORMAT SPOTTY HOME SWEET HOME), (... | 0.001383 | 0.002075 | 0.001383 | 1.000000 | 482.0 | 0.001380 | inf |
| 38067 | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... | 0.002075 | 0.001383 | 0.001383 | 0.666667 | 482.0 | 0.001380 | 2.995851 |
| 38068 | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... | 0.002075 | 0.001383 | 0.001383 | 0.666667 | 482.0 | 0.001380 | 2.995851 |
| 38069 | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... | 0.002075 | 0.001383 | 0.001383 | 0.666667 | 482.0 | 0.001380 | 2.995851 |
| 38070 | ((Quantity, DOORMAT MERRY CHRISTMAS RED), (Qua... | ((Quantity, DOORMAT UNION FLAG), (Quantity, DO... | 0.001383 | 0.002075 | 0.001383 | 1.000000 | 482.0 | 0.001380 | inf |

In [17]:

*# We can try below products in the loop*

*# ('Quantity', 'VANILLA SCENT CANDLE JEWELLED BOX')*

*# ('Quantity', 'DOORMAT MERRY CHRISTMAS RED')*

*# ('Quantity', 'DOORMAT RESPECTABLE HOUSE')*

*# ('Quantity', 'DOORMAT SPOTTY HOME SWEET HOME')*

*# ('Quantity', 'DOORMAT UNION FLAG')*

*# ('Quantity', 'DOORMAT UNION FLAG')*

recommendation\_list = []

for i, product **in** sorted\_rules["antecedents"].items():

for j **in** list(product):

if j == ('Quantity', 'MARIE ANTOINETTE TRINKET BOX GOLD'):

recommendation\_list.append(list(sorted\_rules.iloc[i]["consequents"]))

#### **As a result, the products that are purchased together with the**('Quantity', 'MARIE ANTOINETTE TRINKET BOX GOLD')**product are seen below.**

In [18]:

recommendation\_list

Out[18]:

[[('Quantity', 'DOORMAT MERRY CHRISTMAS RED'),

('Quantity', 'DOORMAT NEW ENGLAND'),

('Quantity', 'DOORMAT HEARTS')]]

**Evaluation:**

Evaluations in market basket insights are used to assess the quality and usefulness of the insights generated. There are a variety of different evaluation metrics that can be used, but some of the most common include:

* Support: Support measures the frequency with which an itemset or association rule occurs in the data. A higher support value indicates that the itemset or association rule is more common.
* Confidence: Confidence measures the probability of purchasing a consequent item, given that the antecedent item has already been purchased. A higher confidence value indicates that the association rule is stronger.
* Lift: Lift measures the strength of an association rule relative to what would be expected by chance. A lift value greater than 1 indicates that the association rule is statistically significant.

In addition to these quantitative metrics, it is also important to evaluate the qualitative aspects of market basket insights. This includes assessing the insights to determine whether they are actionable and consistent with business knowledge.

Here are some examples of how evaluations can be used in market basket insights:

* A retailer could use support to identify the most popular product bundles.
* A grocery store could use confidence to identify which products are most likely to be purchased together.
* An e-commerce company could use lift to identify which product recommendations are most likely to lead to sales.

By using evaluations, businesses can ensure that they are using the most valuable market basket insights to improve their performance.

Here are some additional tips for evaluations in market basket insights:

* Use a variety of evaluation metrics. Different evaluation metrics can provide different insights into the quality and usefulness of market basket insights. By using a variety of metrics, businesses can get a more complete picture of the value of their insights.
* Consider the business goals. The evaluation metrics that are most important will depend on the specific business goals of the market basket analysis project. For example, if the goal is to identify the most popular product bundles, then support will be the most important metric.
* Use domain knowledge to inform the evaluation. Business knowledge can be used to assess the reasonableness and actionability of market basket insights. For example, if an association rule suggests that customers who purchase milk are more likely to purchase diapers, this is consistent with business knowledge and is therefore more likely to be actionable.

By following these tips, businesses can use evaluations to identify the most valuable market basket insights and improve their performance.

**CONCLUSION:**

Market basket insights are a valuable tool that can be used to gain insights into customer behaviour and improve business performance. By understanding what products are often purchased together, businesses can develop targeted marketing campaigns and promotions, improve product placement, and reduce costs