MARKET BASKET ANALYSIS (MBA) USING CLUSTERING AND FP GROWTH ALONG WITH ASSOCIATION RULES TO SUGGEST DATA DRIVEN-MARKETING STRATEGIES

Declaration

Include the following statement on the same page as the acknowledgments. I declare that I have personally prepared this report and that it has not in whole or in part or been submitted for any other degree or qualification. Nor has it appeared in whole or in part in any textbook, journal or any other document previously published or produced for any purpose. The work described here is my/our own, carried out personally unless otherwise stated. All sources of information, including quotations, are acknowledged by. means of reference, both in the final reference section and at the point where they occur.

Acknowledgement

Firstly, I would like to express my deepest gratitude to Almighty Allah for providing me with the strength, knowledge, and wisdom to complete this dissertation.

I am immensely grateful to my parents, whose love, guidance, and sacrifices have provided me with the foundation upon which this work is built. Their unwavering faith in my abilities has been a constant source of motivation and resilience throughout this journey. To my wife, I extend my heartfelt thanks for her patience, understanding, and unwavering support. Her encouragement and faith in me, even in the most challenging times, have been invaluable. Her love and companionship have made this journey enjoyable and fulfilling. I would also like to acknowledge the invaluable contributions of my academic mentors, Supervisor and colleagues specially Yusuf Makhbool vai and Hassan, whose insights and expertise have greatly enriched this work. Their constructive feedback and encouragement have been instrumental in shaping this dissertation.

Finally, I would like to express my gratitude to all those who have directly or indirectly contributed to this work. Their support has been invaluable, and this dissertation would not have been possible without them.

Contents

Abstract

This paper explores the application of Market Basket Analysis (MBA) in the e-commerce domain, leveraging data mining techniques to uncover patterns in consumer purchasing behaviour. Through a detailed examination of transactional data, the study employs the k-means clustering and then the Apriori algorithm, FP-Growth algorithm to identify frequent item sets and association rules, providing insights into consumer preferences and behaviour. The analysis reveals significant associations between products, offering valuable implications for product placement, promotional strategies, and personalized marketing, cross-selling with data driven strategies. The findings emphasize the potential of Market Basket Analysis (MBA) to inform strategic decision-making in e-commerce, enhancing customer satisfaction and operational efficiency. The study contributes to the evolving landscape of e-commerce strategies, highlighting the importance of data-driven approaches in understanding and catering to consumer needs.

1 Introduction

Background of the Study

The e-commerce industry is an ever-changing and intensely competitive sector. Businesses operating in this field are constantly searching for ways to boost sales and improve customer satisfaction. One effective method is the implementation of sales promotions and discounts on specific products, crossselling, customized packages and niche promotion. These tactics not only attract new customers but also encourage repeat purchases, ultimately driving overall sales. However, the success of these strategies heavily relies on their timing and alignment with customer needs and preferences, underscoring the importance of identifying sales patterns. Sales patterns yield valuable insights into customer behaviour, encompassing their buying tendencies, preferences, and receptiveness to promotions. By acknowledging these patterns, businesses can adjust their promotional strategies, optimizing the application of discounts with greater precision and impact. Yet, identifying and leveraging these patterns poses a formidable challenge. This process entails analysing extensive sales data, a task that can be complex and time-consuming.

Problem Statement

Applications of data mining techniques, such the Clustering, FP-Growth, Apriori algorithm, to sales promotions and discounts are still largely unexplored, despite their ability to analyse sales data and find patterns, seasonal trends that can guide promotional campaigns. This study is conducted to minimize the gap.

- To analyse the provided dataset to identify product clusters, frequent items and association rules among products within an e-commerce setting.
- To interpret the findings from the Market Basket Analysis(MBA) to gain insights into consumer purchasing patterns, preferences, when the sales increases and behavioural pattern changes
- To evaluate the impact of these insights on strategic decision-making for e-commerce businesses, particularly in the context of product assortment, cross selling and personalized marketing.

As we explore into the details of Market Basket Analysis(MBA), our aim is to contribute to the ongoing evolution of e-commerce strategies, ensuring businesses can harness the power of data to enhance customer satisfaction and optimize operational efficiency.

Significance of the Study

This study is important because it might improve how well discounts and sales promotions work. This research offers a data-driven approach to promotional marketing that can help organisations boost sales and enhance customer happiness. The results of this study can also add to the corpus of information already available in the field of sales and marketing, especially when it comes to ecommerce. This study has a wide range of practical applications. E-commerce companies may use the results to improve their marketing tactics and become more competitive in the market. Additionally, other academics and practitioners who are interested in investigating the application of data mining techniques in sales and marketing might utilise the study as a reference. The ultimate purpose of this study is to increase sales and consumer happiness in the e-commerce industry by offering a thorough and data-driven approach to sales promotions and discounts. This study aims to reveal the unrealized potential of data-driven advertising methods by identifying sales patterns and utilising data mining tools.

Structure of the report

The goal of the research study is to thoroughly examine the use of Market Basket Analysis(MBA) in ecommerce, with an emphasis on understanding customer behaviour, buying patterns, The research begins with an introduction describing the goals and providing background information. Then, a thorough literature study explores several topics, such as consumer behaviour, Market Basket Analysis(MBA), and the use of data mining techniques to transactional data analysis. The study's importance is emphasised throughout the report, especially in light of revenue strategy optimisation. The study then discusses the methodology, including data collecting, pre-processing, and analytical tools, with a particular emphasis on the widespread application of the Clustering, FP-Growth, Apriori algorithm. The analytic results are presented in the following sections, with a focus on frequent item sets, association rules, consumer behaviour insights, and the financial success of different product categories, trend analysis. The conclusion offers advice for e-commerce strategies, briefly summarises the key findings, and identifies possible directions for further investigation. The technical details of the methods utilised, additional tables and figures, and references are all included in good proportion in this work. This methodical technique ensures that the research topic is thoroughly investigated in a systematic and complete manner, improving accessibility for a clear understanding of the study's results and their wider implications.

2 Literature Review

Market Basket Analysis(MBA)

Market Basket Analysis(MBA) (MBA) is a collection of statistical affinity calculations that assist managers in better understanding the purchase trends for their business (Ünvan & Yüksel, 2021). According to (Chen, et al., 2005) , Market Basket Analysis(MBA) is computationally efficient and mentioned that the strategy works better in large item sets of the store. The study by (Setiabudi, 2011) is an example of the practical application of Market Basket Analysis(MBA) (MBA) in understanding consumer behaviour and establishing marketing strategies to increase sales and profits. (Marcos, et al., 2021) has mentioned In the context of Market Basket Analysis(MBA), a transaction is any item that a customer buys all at once. In other words, every entry in the dataset includes every item a client has ever bought. A procedure groupings Individual items—such as the sales ID or invoice—to be examined by an identifier that shows the event or transaction. (Moodley, et al., 2019) mentions about the intensified competition in the shopping sector, evolving spending habits, and increased price sensitivity, specifically after the 2008 recession in the UK and emphasized the consequence of understanding items frequently bought together and the measures used to determine extensive support and confidence in Market Basket Analysis(MBA). In addition to finding intriguing results, Differential Market Basket Analysis(MBA) can also solve the issue of a possibly large number of

irrelevant results. Results are compared in this analysis between various stores, between consumers in various demographic categories, between seasons of the year, between days of the week, etc (Ünvan & Yüksel, 2021).

Market Basket Analysis(MBA) in E-commerce.

In the era of cutting-edge technology, people want to reduce time chain of what they are doing. Specifically, after Covid-19 people has become more accustomed to ecommerce. (Nakano, 2023) mentions that recent research has discovered that e-commerce can conduct to customer demand concentration on common products under specific situation, such as recommendations of popular products and past-purchase shortcuts. This research focuses on multichannel grocery retailing in Japan, found that compared with demand in physical stores, demand online is more concentrated on popular products. Moreover, the online shopping experience has a controlling effect, resulting in greater demand concentration among more experienced customers. These results highlight the nuanced nature of the long tail effect in e-commerce and its impact on customer purchasing behaviour, particularly in the context of multichannel retailing and product assortment strategies. The long tail effect in e-commerce denotes to a business strategy that focuses on selling numerous niche products with low demand, as opposed to only selling a reduced number of popular items.

The study by (Rangga, et al., 2022), indicates that there is a 50–60% probability that a client who purchases a certain product would also purchase a related product. With this data, one can optimise product placement to boost sales and develop successful cross-selling and upselling tactics. However, there are a few disadvantages to the Market Basket Analysis (MBA) in e-commerce, though. One of the main disadvantages is the need for large data quantities. Market Basket Analysis (MBA) requires a substantial volume of transaction data to identify meaningful patterns and connections. This would not be feasible for smaller e-commerce businesses that do not have access to such extensive statistics. The Market Basket Analysis (MBA) results are significantly impacted by the quality of the data as well. If the transaction data is inaccurate or insufficient, the analysis could yield misleading results. Another flaw in the Market Basket Analysis (MBA) is its disregard for the purchase order. Just the relationships between the things are shown, not the order in which they are purchased. This might make it less accurate in predicting the buying patterns of its clients.

In a research study by (Xudong, et al., 2024) mentions 20% of all retail sales worldwide will occur online by 2022 thanks to the rapidly expanding e-commerce sector. Retail e-commerce sales worldwide hit $5.5 trillion in 2022 and are projected to grow to $7.3 trillion by 2025. With projected revenues of $780 billion in 2022, the Chinese e-commerce behemoth Alibaba is the biggest online retailer globally. In modern e-commerce landscapes, customer experience emerges as a pivotal factor shaping

consumers' perceptions of value and service quality, directly influencing their loyalty. The holistic nature of this experience underscores the importance of efficient communication between clients and online shops, where the integration of chatbots in customer support is increasingly prevalent. The growing importance of eco-conscious practices, which prioritise sustainability and environmental preservation, is a major trend for 2023. Resource conservation, e-waste management, and the zero-waste movement have all gained significant traction as customer demand for environmentally friendly business practices rises. In the future, voice assistants are expected to handle 45.4% of all searches in the US by 2026, as voice search usage is predicted to drastically change digital interactions. This change emphasises how important it is for companies to modify their online platforms in order to accommodate this changing nature of search. The identified trends aren't verified by case studies or empirical data in this paper. The study's foundation is an examination of specialised reports and professional literature, which can have limited the analysis's depth. Additionally, the study makes no mention of the difficulties or obstacles that might arise when implementing the trends that have been identified in the e-commerce sector.

Consumer Behaviour and Purchase Patterns in ecommerce

Consumer purchase behaviour is a complex process influenced by numerous factors, including economic, social, and political stability. The steadiness of the surroundings can significantly influence consumer decision-making, specifically during periods of uncertainty. For instance, there is a huge change in the purchase behaviour before and after Covid-19.

The paper ‘’Online Consumer Typologies and Their Shopping Behaviours in B2C E-Commerce Platforms’’ by (Anon., 2019) focuses on consumer segmentations, which were found to exhibit distinctive online shopping behaviours, suggesting that a broad online consumer audience is actually a collection of various consumer segments with unique behavioural traits rather than a single market segment. According to (Davis, 1989), The Technology Acceptance Model (TAM) a broadly used technique in information systems works, it identifies several factors that influence users' decision to adopt a new technology. The model has been commenced to be applicable in understanding users' attitudes and behavioural objectives toward adopting new technologies.

The study of (Anon., 2019) designed a behaviour valuation model to evaluate each segment's purchasing behaviours. In order to be assured that the relationships between the model constructs are strong throughout the four segments, this model was used to test the invariance of causal paths across all segments. The four segments are as follows, *Shopping Lovers*, consumers with a strong interest in online shopping and positive attitudes and behaviours towards e-commerce activities

Consumers who make direct online purchases and demonstrate a high level of confidence and trust in e-commerce platforms. *Suspicious Browsers*, consumers who spend time on various online platforms but show reluctance to make online purchases due to high levels of distrust toward the Internet, concerns about privacy and security during online transactions, as well as worries about the delivery and return process of online products. *Incompetent Consumers,* who struggle with online shopping activities and exhibit low confidence and competence in using e-commerce platforms. The findings determined that consumers across all segments develop positive attitudes regarding online shopping as they believe it to be beneficial and coherent with their lifestyle. Furthermore, consumers across all demographics may foster favourable opinions on online shopping because of pressure to shop online from significant others like friends, family, and the media (G. & A, 2016).

Limitations of the case study (Anon., 2019) are, firstly, the study's participants were limited to university students living in a certain area. It is advised that equivalent analyses be conducted in future investigation with individuals who are not students and over a greater geographical region in order to improve the generalizability of the study findings. Secondly, this study's segmentation analysis and model testing methods rely on the respondents' self-reported data.

The study conducted by (Mansour, 2020) from March 15, 2020, to May 25, 2020, investigated the impacts of COVID-19 on the five largest e-commerce companies in the world: Amazon, Alibaba, Rakuten, Zalando, and ASOS, which measured the frequency of the virus through swelling infections and deaths, as well as new cases and deaths on a daily basis. The research determined on the response of the global e-commerce market to the effect of the virus, evaluated by the daily gains of the shares of these companies to the universal financial markets. The results of the investigation indicated that the companies attained positive daily returns on average. The analysis also discovered that the frequency of COVID-19 had varying influences on the companies, with the most effective variables being growing infections and deaths, as well as new cases. The percentage of the impact of the virus spread varied from one firm to another, depending on the country to which it belonged. For example, Amazon and ASOS were most influenced by growing infection cases, consistent with these countries being the most influenced by the virus during the research period. On the other hand, Alibaba and Rakuten were most influenced by new cases, and Zalando was most influenced by increasing deaths. This research presents valuable insights into how the COVID-19 pandemic has impacted the global ecommerce market and the changing effects on major e-commerce companies based on different countries and regions.

Research findings by (Riccardo, et al., 2015) shows there is a growing trend of environmental sustainability has become a significant focus in the e-commerce industry. Over the past decade, there

has been a significant shift from simply recognizing the environmental effects of e-commerce to a more measurable evaluation of these effects. While some sectors, such as books and groceries, have been addressed, other promising areas like clothing and consumer electronics have only been partially considered. The influence of e-commerce on the environment is a key consideration, and as the industry continues to grow, it is essential for organizations to prioritize sustainability and take important steps to reduce their environmental footprint.

In this modern era, AI is integrating all the platforms of the Business. AI in e-commerce has made it easier to understand and predict consumer behaviour, accompanying traditional analytics. AI enables the analysis of larger and diverse datasets, allowing for more targeted and personalized marketing, which can lead to improved customer engagement, advanced conversion rates, and enriched customer satisfaction. However, the study by (Saikat, et al., 2020) mentions the limitations in demonstrating the behavioural patterns specifically due to the humungous quantity and alteration of data paving the pathway for AI to enter the situation as coupling with Analytics it can release its values to make easier.

Digitalization is used to describe how digital technologies can be used to alter a business model and create new chances for revenue generation and value creation. It is the process of converting a company to operate online (Anon., n.d.). A research study by (Petra & Petra, 2020) reflects on Demand has increased significantly as a result of the Czech economy's expansion; in fact, the global retail ecommerce market saw a threefold increase in sales from USD 1.336 billion in 2014 to USD 3.453 billion in 2019. According to the report, there are 1.92 billion online shoppers worldwide, the bulk of whom make cross-border transactions.

The study is organised into four main sections: methodology, which includes details on sample and statistical procedures, results, which include conclusions from a marketing perspective, and conceptual approaches to explain online purchasing behaviour. The theoretical section uses secondary data analysis and a questionnaire to introduce both online and offline payment methods. According to reports, Czech e-commerce generated almost 160 billion in revenue in 2019, or 10% of the country's total retail market, with 44,000 businesses operating in the country. Furthermore, the study's primary conclusions show that the percentage of people over 16 who make purchases online is consistently increasing, rising from 34.4% in 2014 to 53.9% in 2018. The age group that is most active is the 25–34-year-old demographic, but the share of all age groups is increasing overall. The proportions of each gender are changing; more and more men and women are making purchases online. According to the study, certain demographics, such as older people and mothers on maternity leave, have a lot of potential for future e-commerce engagement. Particularly during their maternity leave, mothers become significant buyers for their whole family. In addition, those with a university degree make up

a large portion of online shoppers—76% of all transactions in 2018 were made by people with a university degree, compared to only 16.5% by people with only a primary education. The dynamic character of e-commerce trends and the possibility of future expansion and diversification are highlighted in the paper's conclusion. Along with the very insightful information's this paper has some limitations. The survey was asked by only fifty people, this suggests that the population is very small that doesn't reflect all the behaviours of the e-commerce buyers.

The study by (Hani A. Jawa, 2015) aimed to understand consumer behaviour towards online shopping in Saudi Arabia, focusing on factors like website quality, perceived trust, convenience, and the impact of advertisements and promotions. Conducted in Dammam city with a sample size of 107 participants, the research utilized a questionnaire and SPSS software for data analysis. The findings indicated a notable age group dominance of 20-25 years and a higher percentage of single and college graduate participants. Among the hypothesized factors, only website quality showed a significant positive impact on online shopping behaviour, while convenience, trust, and promotional strategies did not significantly influence consumer decisions. Pros of this study include a focused examination on a region with growing e-commerce potential and the identification of website quality as a crucial factor for online shopping behaviour, providing valuable insights for e-commerce platforms aiming to enhance user experience. However, the study's limitations include a small sample size and regional focus, potentially affecting the generalizability of the findings. Future research could benefit from a broader demographic to gain more comprehensive insights into consumer behaviour across Saudi Arabia. In summary, this research contributes to understanding the dynamics of online shopping in Saudi Arabia, highlighting the paramount importance of website quality in influencing consumer behaviour, while suggesting that factors like perceived trust, convenience, and promotions may not play as significant a role in this region's e-commerce landscape (Hani A. Jawa, 2015).

Data Mining Techniques for Transactional Data

A research study by (Santoso & M., 2021) reflects on data is now being accumulated in large databases due to the progress of information technology, which has drastically changed the way organisations compete. To remain competitive in the face of this, businesses must come up with creative solutions. Data mining emerges as a critical technique to enable the extraction of important knowledge from these massive data sources and enhance corporate decision-making by transforming this huge data into actionable insights. Data Mining involves the extraction of useful information from datasets, utilizing a range of algorithms and methodologies from disciplines like statistics, machine learning, mathematics, and artificial intelligence (Ridwan, et al., 2013).

In this paper (Santoso & M., 2021) concludes, with 40% support and 80% confidence, the Apriori algorithm successfully detects buying patterns for developing sales bundles, such as the common pairing of toothpaste and detergent. Higher minimum support and confidence, however, result in fewer itemsets and rules, which has an impact on accuracy.

(Setiabudi, 2011) has conducted research on Market Basket Analysis (MBA) using apriori algorithm to discover common sets of items in transaction history and the database. The picked item sets exceeding the minimum support value threshold were used to generate association rules, and the confidence was processed using hybrid dimension association rules. The mining process outcomes revealed the correlation among association rules and the confidence that can be analysed. This research illustrates the use of Market Basket Analysis (MBA) MBA in a real-world setting, showcasing how the Apriori algorithm can be employed to gain insights into consumer purchasing patterns and drive business decisions.

(Nanang, et al., 2019) in his study of "Analysis of Customer Purchasing Patterns on E-Commerce Transactions Using Apriori Algorithms and Sales Forecasting Analysis with the Weighted Moving Average (WMA) method" yielded several significant conclusions. the Apriori algorithm demonstrated a higher accuracy value of 130.75, outstanding the FP-Growth Algorithm, which scored 120.63, suggesting its efficacy in producing better association rules based on the strength of the rules (lift ratio). Secondly, the Apriori Algorithm was able to produce recommendations for related product categories based on association rules, serving as a source for product categories to promote. Thirdly, the application of the Apriori algorithm in the form of a system prototype met the standard value of the company's software quality assurance (SQA) testing and achieved 100% verification of the prototype of the rapid miner application, using a minimum support of 2% and minimum confidence of 90%. Lastly, the study found that sales forecasting using a weighted moving average (WMA) model with 3 weights, based on the association rules of the Apriori algorithm, could be implemented to set sales targets, with a focus on achieving a desired percentage increase in sales and maintaining MAPE error levels below 2.0%.

An analysis with a convenient store by (Aldino, et al., 2021) that it has faced a decline in its sale after Covid-19. They have used this data set to compare two datasets to compare between two algorithms. According to the study, the FP-Growth algorithm performed faster and more accurately than the Apriori method. In about 6 seconds, FP-Growth generated 19 rules and combined 3 itemset with an accuracy of 217% and a rule strength of 112.66%. The Apriori algorithm, on the other hand, created six rules in 30 seconds and combined three things with a rule strength of 52.47% and an accuracy of

46%. The FP-Growth method is a superior option for analysing huge datasets to identify consumer purchasing trends since it is more efficient than the Apriori technique. After the analysis, the store changed their item placement which boosts their performances.

The paper by (Raorane, et al., 2012) investigates into the application of Market Basket Analysis (MBA) (MBA) using association rule mining to understand customer behaviour and enhance decision-making in retail environments. The authors emphasize the pivotal role of data mining tools in analysing vast amounts of transactional data to unearth patterns that can inform strategic business decisions. By implementing Market Basket Analysis(MBA), they aim to leverage consumer behaviour insights to gain a competitive edge in the marketplace.

The study draws the process of knowledge discovery from databases, highlighting the steps from data cleansing to knowledge presentation, with a particular focus on the extraction of data patterns through data mining. The research employs a dataset from Shetkari Bazar, a supermarket in Maharashtra, India, to demonstrate the methodology. The authors analyse customer transactions to identify frequently purchased item sets, applying metrics such as support and confidence to evaluate the strength and reliability of the associations between items. The findings discover significant associations among goods purchased together, suggesting strategies for product placement and promotions to enhance sales and customer satisfaction. The paper concludes that data mining tools, particularly MBA, are effective in revealing purchasing patterns, offering valuable insights for retail management and strategic planning. This paper contributes to the literature on MBA and association rule mining by providing a practical application of these techniques in a retail context, showcasing their potential to transform raw transactional data into actionable business strategies. The study underscores the importance of understanding customer purchasing behaviour and the role of data mining in facilitating informed decision-making in the competitive retail sector (Raorane, et al., 2012).

A study by (Young-Chan, et al., 2018) explores the application of data mining techniques, specifically K-means clustering and association rules, for customer segmentation and Market Basket Analysis(MBA) within the context of a Korean retailing company's big data. The study aims to address the challenges faced by small and medium-sized enterprises (SMEs) in South Korea, which often struggle with customer relationship management (CRM) due to a lack of technological innovation and data analysis capabilities. The authors propose a method to segment customers into VIP and non-VIP groups using the RFM (Recency, Frequency, Monetary) model and then apply K-means clustering to further analyse VIP customers' transaction data. The study looked at 1866 customers and identified 227 of them as VIPs using the RFM (Recency, Frequency, Monetary) model and K-means clustering to break down their purchasing patterns. They then used the Apriori algorithm to find patterns in the

buying habits of these VIP customers, setting the bar at 10% for the least amount of support and 85% for the minimum confidence level needed to consider a pattern significant. They found 29 patterns that met these criteria, and out of these, 10 had a confidence level above 90%. One of the strongest patterns discovered was that customers who buy children's products and beauty items often also buy women's clothing. This pattern was seen in 44.611% of the cases and had a lift of 1.243, indicating a significant association. This study concluded that home products and women's wear are frequently associated with food purchases, suggesting that the purchase of home products or women's wear mainly affects the purchase of food (Young-Chan, et al., 2018).

The research by (Soma, et al., 2021) showed that using Principal Component Analysis (PCA) and KMeans clustering methods allowed for effective grouping of customers and products in the online clothing market. They looked at things like how much money customers make each month and how much they're willing to spend on items. From the K-Means clustering, they found four different types of customer groups based on their income and how much they spend. For example, they found a group of customers with lower incomes who tend to buy less expensive items and another group with higher incomes who buy pricier items.

When it came to dividing up the products, they also used K-Means clustering, this time focusing on how much was sold and how much money those sales made. This helped identify which products sold well and made a lot of money, versus those that didn't sell as much or were expensive but didn't sell well, impacting their revenue.

The study also tested how good their recommendation system was by looking at what customers bought over a certain time and seeing how many of those purchases matched the system's recommendations. They found that a lot of the products bought were ones the system had suggested, showing that the system was pretty good at figuring out what customers liked and helping them find products that matched their taste.

This work by (Mustapha, et al., 2015) portrays the application of data mining in e-commerce, presenting both its advantages and challenges. It emphasizes the significance of utilizing vast amounts of structured and unstructured data, termed "big data," which e-commerce platforms amass from various sources. Through the implementation of data mining processes, including association, clustering, and prediction algorithms, e-commerce companies can enhance merchandise planning, sales forecasting, customer relationship management, and market segmentation. The benefits outlined include improved decision-making through the analysis of customer purchasing behaviours, sales forecasting, and effective market segmentation, ultimately leading to increased competitiveness and revenue generation for e-commerce companies. However, the paper also acknowledges the

challenges faced in data mining within e-commerce, such as the need for accurate spider identification, data transformation complexities, scalability of algorithms, making models comprehensible to business users, supporting slowly changing dimensions, and making data transformation and model building accessible to non-technical users. In summary, while data mining presents a valuable tool for e-commerce companies to leverage customer data and enhance their services, it also brings forth technical and operational challenges that need to be addressed to fully realize its potential (Mustapha, et al., 2015).

According to (Arcos, et al., 2019)the efficiency of the Apriori Algorithm for association rule mining in online transaction data analysis, specifically in the context of Misumi Philippines. The researchers aimed to improve the process of identifying frequent item sets for marketing strategies like product bundling or promotions. By modifying the Apriori Algorithm with a transaction reduction approach and utilizing hash tables, the study achieved a significant reduction in runtime, thereby increasing the performance of data mining tasks. This study enhanced Apriori Algorithm demonstrated a notable improvement in generating association rules, with a significant decrease in runtime compared to the original algorithm. The market basket analysis conducted as part of the study revealed potential product bundles that could be offered to customers, based on the frequent item sets identified through the mining process. These results were validated using a Java application simulation on the Misumi Philippines dataset, which consisted of various attributes like product codes, names, quantities, and prices. The study acknowledged certain limitations, including the reliance on manual preprocessing of data, which is susceptible to human errors. Additionally, the research was confined to market basket data, suggesting that other areas and types of data could also be explored for association rule mining. The authors suggested that future work could include developing an application that automatically determines and generates optimal product packages based on the simulation results, thus addressing some of the manual aspects of the current methodology.

4 Methodology

Research Design:

The study is descriptive analytics where I've used, python programming. I've used data mining technique called Clustering and Market Basket Analysis (MBA) to figure out the trends of why the customer behaviour is changing over time. The figure by (Ünvan & Yüksel, 2021) describes what are the steps that could be taken for datamining following towards clustering and Association.

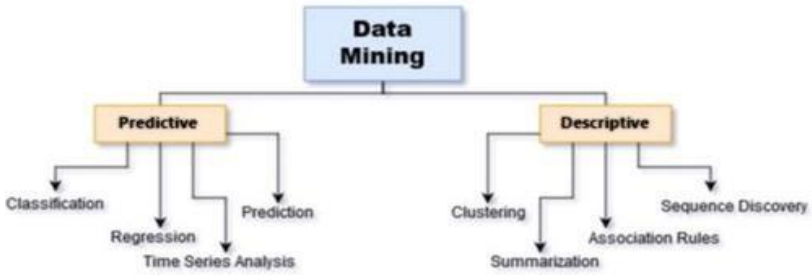


Figure Data Mining Flow Table

These methods are used to identify patterns of items frequently bought together, which helps in creating effective promotions and discounts, item placement and boost cross selling. The main models used for this analysis are Clustering, the Apriori and FP-Growth algorithms with association rules, that provides a solid foundation for identifying purchasing patterns and informing strategic decisionmaking related to promotions and discounts and demonstrate according. These algorithms are used to identify frequent item sets, which are sets of items that appear together in the transaction data with a frequency above a certain threshold. The research flow chart and data description is given below. is given below.

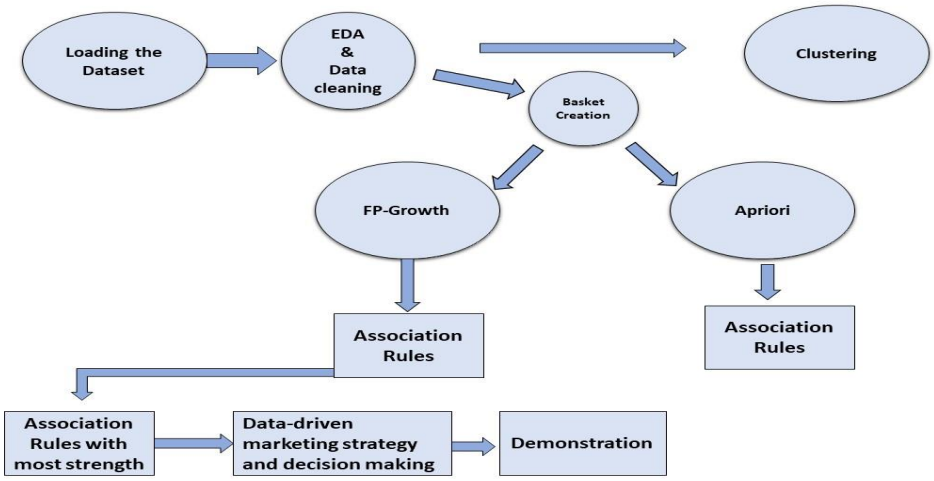


Figure: Research Flow Chart

Name	Description
Purchase Date	**Date and Time of Purchase**
Transaction	**Purchase Records**
Product	**Product Specification**
Product Category	**The category to which product belongs**
Quantity	**Amount Purchased for a unique Product**
Product Revenue	**Revenue for a particular product**

Table: Data description

Data Collection:

The data for this research paper was collected from a Kaggle dataset, which supplies a thorough set of transaction records from an ecommerce platform. The dataset includes Product, Product Category, Transaction, Quantity and Product Revenue. This large and thorough dataset is a brilliant resource, I've found for Market Basket Analysis (MBA). This dataset yields informative market basket research results that will be used to utilise and create effective marketing campaigns, understand consumer purchasing patterns along with time , and make informed business decisions. The accessibility of this data from a reputable source such as Kaggle enhances the credibility and reliability of the analysis. The use of real-world operation data from a large and diverse set of purchases further supports the relevance and applicability of the findings to practical business scenarios. In addition to the transactional data, the following paper also incorporates exploratory data analysis (EDA) and data cleaning processes to ensure the dataset is well-prepared for Market Basket Analysis (MBA). These preparatory steps are essential for ensuring the accuracy and reliability of the subsequent analysis which reflects its practical efforts in ecommerce retail settings.

Exploratory Data Analysis and Data Pre-processing:

Exploratory Data Analysis (EDA) and data pre-processing are essential steps in any data analysis project, including Market Basket Analysis (MBA). They involve recognizing the nature of the data, cleaning it, and preparing it for analysis. Here the dataset consists of transaction data with various attributes such as product name, product category, Transaction, Quantity, and product revenue. The initial dataset consists of transaction data with various attributes such as Product, Product category, Transaction, Quantity, and product revenue. The EDA process also involves visualizing the data to gain insights. For instance, top 20 products has been identified which has been purchased and generated huge revenues. The dataset is first examined to understand its structure, data types, and basic statistical properties.

Data-cleaning involves cleaning and transforming the data to make it suitable for analysis. In this study, the data cleaning involves steps include handling missing values and removing unnecessary data. For instance, any rows with missing Transactions are dropped from the dataset. This is important because Transactions are crucial for Market Basket Analysis(MBA), as they link together items bought in the same transaction. Another cleaning step involves make sure that all values in the 'Quantity' column are greater than zero. This is because a zero value in this column would not make sense in the context of transaction data - if an item appears in the transaction data, it must have been purchased at least once. Finally, the 'Product Revenue' column, which is primarily an object data type, is converted to a numerical data type. This is necessary because mathematical operations cannot be performed on object data types, and numerical data types are suitable for statistical analysis.

In conclusion, EDA and data pre-processing are essential steps in preparing the data for Market Basket Analysis (MBA). It helps in understanding the data, identifying, and handling anomalies, and transforming the data into a suitable format for analysis.

Data Preparation/Feature Engineering

The main objective for the data preparation was to convert unstructured transactional data into a format that is appropriate for Market Basket Analysis (MBA). To associate the data with the requirements of Market Basket Analysis (MBA) algorithms, the process involving of several necessary steps, each with a distinct function. To group together the total quantities of each product within individual transactions, we first grouped the data by products and transactions. To effectively capture the essence of customer purchasing patterns at the transactional level. Then pivoted the data to transform it into a matrix format. In this matrix, each row represents a unique transaction, and each column signifies a different product, with cell values reflecting the total quantity of each product in the respective transactions. This step was imperative for converting the transactional data into a binary format, a prerequisite for effective analysis. Data Frame’s index to a default integer index for simplicity and better data handling has been reset. Single-item transactions provide only a limited amount of information about purchasing patterns. Therefore, the dataset is restricted to only contain transactions in which more than one product were purchased. The mentioned filtering performed a crucial role in guaranteeing that the ensuing analysis was concentrated on significant transactional data that could disclose noteworthy item associations.

The data preparation methodology was carefully designed to ensure the dataset was optimally structured for Market Basket Analysis (MBA). This involved transforming the data into a format that allows for the identification of associations between products, a fundamental aspect of understanding customer purchasing behaviour in retail contexts.

Application of Clustering

Data Aggregation for Clustering

The initial phase of our analysis involved a comprehensive aggregation of transaction-level data. We aimed to consolidate the data by summing up the quantities and product revenues associated with each transaction. This step was crucial for simplifying the dataset to a more manageable form, allowing for a clearer analysis of transactional trends and patterns. Following the aggregation, we embarked on a feature engineering process to prepare the data for clustering. This involved standardizing the dataset to ensure that the scales of the variables did not bias the clustering algorithm. Standardization was achieved by transforming the features (quantity and product revenue) to have a mean of zero and a standard deviation of one. This pre-processing step is essential for distance-based algorithms like K-Means, as it ensures that each feature contributes equally to the distance calculations.

Optimal Cluster Determination: To uncover the inherent groupings within the transaction data, we employed the K-Means clustering algorithm. A critical aspect of this approach was determining the optimal number of clusters. For this purpose, we utilized the Elbow Method, a heuristic used in determining the number of clusters in a dataset. The method involves plotting the within-cluster sum of squares (WCSS) against the number of clusters and identifying the "elbow" point where the rate of decrease sharply changes. This point is considered to represent the optimal number of clusters, as adding more clusters beyond this point does not provide significant improvement in within-cluster cohesion.

Cluster Analysis and Interpretation: With the optimal number of clusters identified, we proceeded to apply the K-Means algorithm with this specific cluster count. Each transaction was assigned a cluster label, indicating its membership to one of the identified clusters. This labelling allowed us to map back the cluster memberships to the original dataset, facilitating a detailed analysis of the transactional data within the context of these clusters. Then conducted an in-depth analysis of the clustered data, focusing on the composition of product types within each cluster. The flow chart enhances more clear view on k-means clustering.

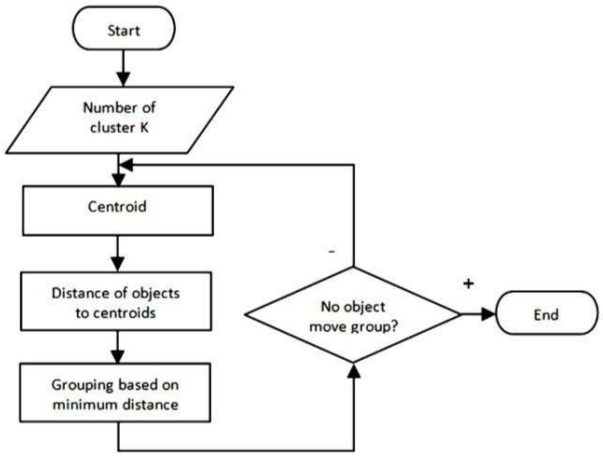


Figure K-means cluster flow chart

This involved grouping the data by cluster and identifying unique products within each group, providing insights into the product mix that characterizes each cluster. Further analysis was conducted to identify the most prevalent products within each cluster. This was achieved by counting the occurrences of each product within the clusters and ranking them to identify the top products. This step provided a granular view of the product preferences or transactional behaviours associated with each cluster, offering valuable insights into the underlying patterns within the transaction data.

Application of Apriori Algorithm.

The Apriori algorithm is a method used in data mining for excavating frequent item-sets and relevant association rules for the determination of frequent itemsets for boolean association rules (Sukenda Wahyu, 2020).

It is designed to operate on databases containing transactions, such as purchases by customers of a store. An itemset is considered "frequent" if it meets a user-specified support threshold. For instance, if the support threshold is set to 0.5 (50%), a frequent itemset is defined as a set of items that appear in at least 50% of all transactions. A full flow through of APRIORI is given below.

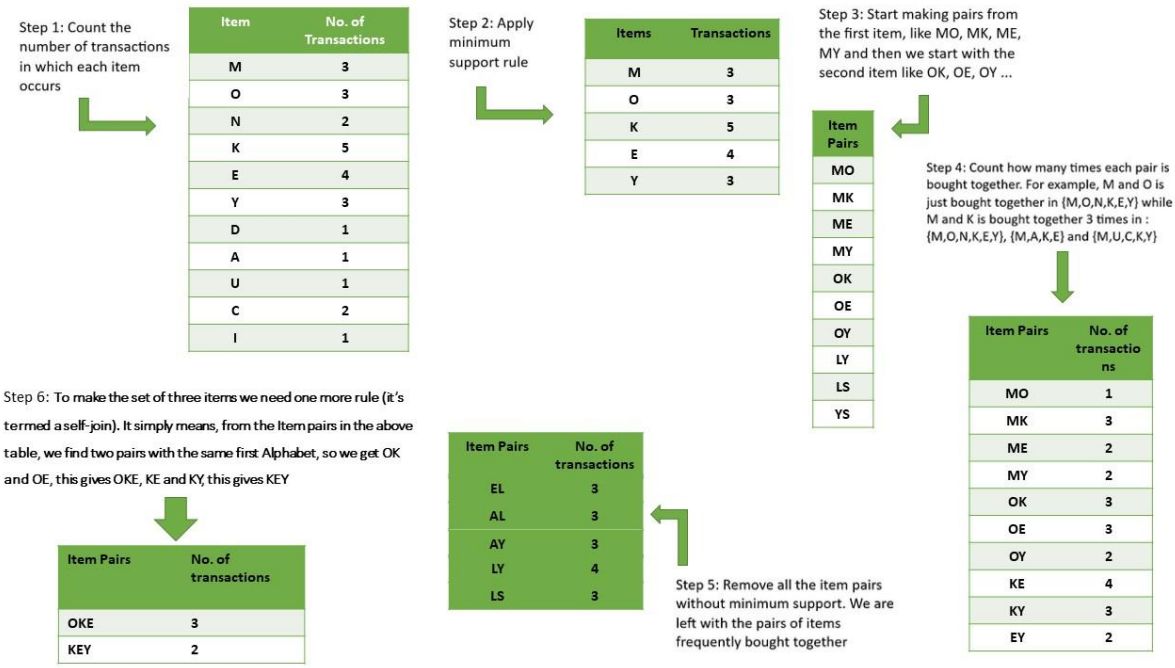


Figure: Apriori Algorithm

The Figure Apriori Algorithm flowchart illustrates a step-by-step process for conducting Apriori algorithm analysis. Here's a detailed explanation of each step:

Step 1: Count the number of transactions in which each item occurs.

This initial step involves scanning the dataset to count how many transactions each item appears in. It's the first pass over the dataset where the frequency of each individual item is tallied.

Step 2: Apply minimum support rule.

Based on the counts from Step 1, this step filters out items that do not meet the minimum support threshold. The minimum support is a user-defined parameter that signifies the minimum number of times an item must appear across all transactions to be considered for further analysis.

Step 3: Start making pairs from the first item, like MO, MK, ME, MY and then we start with the second item like OK, OE, OY... Now, pairs of items are formed to establish which items are frequently purchased together. This is done by pairing each item with all other items that have passed the minimum support threshold. Each pair represents a potential association.

Step 4: Count how many times each pair is bought together. For example, M and O is just bought together in (M, O,N,K,E,Y) while M and K is bought together 3 times in: (M,O,N,K,E,Y), (M,A,K,E) and (M,U,C,K,Y)

The dataset is scanned again to count the frequency of each item pair. This identifies which item combinations are popular among customers.

Step 5: Remove all the item pairs without minimum support. We are left with the pairs of items frequently bought together.Like Step 2, this step filters out item pairs that do not meet the minimum support threshold. This leaves only the item pairs that are considered frequent enough to be meaningful for generating rules.

Step 6: To make the set of three items we need one more rule (it's termed a self-join). It simply means, from the item pairs in the above table, we find two pairs with the same first Alphabet, so we get OK and OE, this gives OKE, KE and KY, this gives KEY

The final step involves extending the analysis from pairs to triplets (or larger sets) of items. This is done using a self-join where item pairs with a common element are joined to form item triplets. The idea is to extend the analysis to find even larger sets of items that are frequently bought together. The flowchart represents the iterative process of the Apriori algorithm, where each step builds upon the before narrowing down the focus to the most relevant item combinations. Each iteration increases the size of the item sets by one, starting from individual items to pairs, and then to triplets, and so on, until no more frequent item sets can be found or the desired itemset size is reached. The Apriori algorithm is efficient because it leverages the principle that all non-empty subsets of a frequent itemset must also be frequent (known as the Apriori property), which allows for significant trimming of the search space.

Application of FP-Growth Algorithm

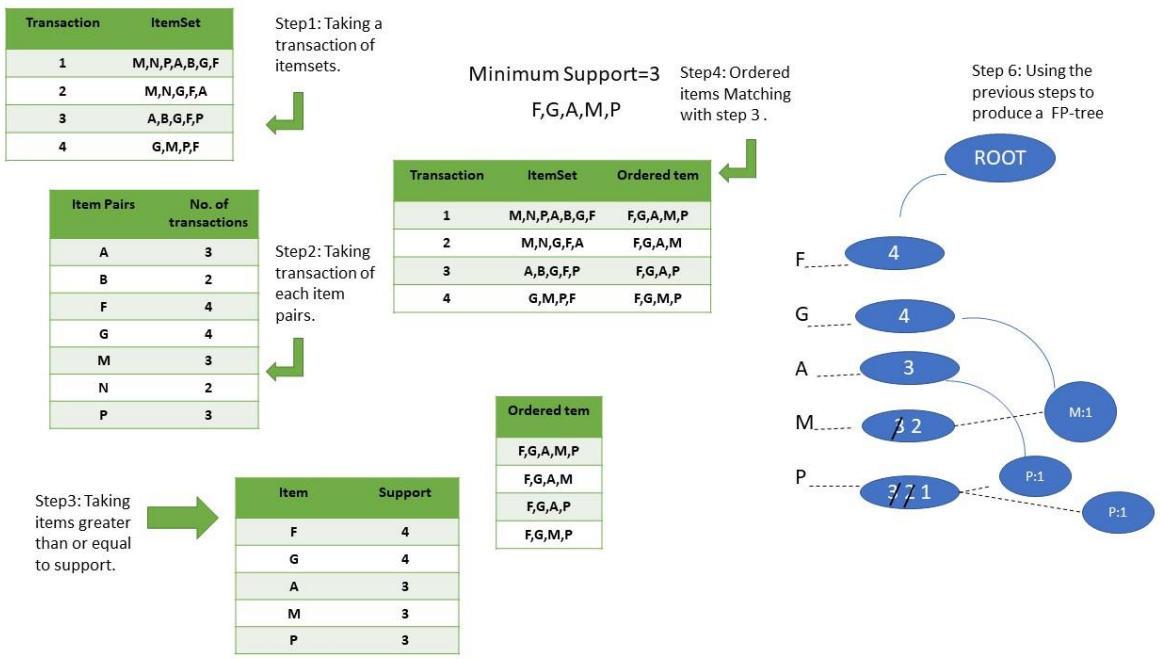


Figure FP Growth

The provided flow chart outlines the process of creating an FP-tree, which is a core component of the FP-Growth algorithm. The FP-Growth algorithm is used for mining frequent itemsets without the need for candidate generation, which is a common bottleneck in algorithms like Apriori. The flow chart describes the steps taken from the initial set of transactions to the construction of the FP-tree. Here's a detailed explanation of each step:

Step 1: Listing the Transactions:

The transactions are listed with their respective itemsets. Each transaction contains a unique set of items bought together.

Step 2: Counting Item Frequency:

The frequency of each item across all transactions is counted. This table lists the items along with the number of transactions they appear in.

Step 3: Applying Minimum Support:

Items are filtered based on a minimum support threshold. In this case, the minimum support is set to 3. Any item appearing in fewer than 3 transactions is discarded.

Step 4: Ordering Frequent Items:

The remaining items in each transaction are ordered by their overall frequency in descending order. This step aligns with the FP-Growth method of processing the most frequent items first.

Step 5: Constructing the FP-tree:

Using the ordered frequent items from the previous step, an FP-tree is constructed. This tree structure starts with a root node, labeled as "ROOT". Each path through the tree represents a set of items. Each node represents an item along with a counter indicating the number of transactions passing through this item. Nodes are created or traversed in the order of the items in the ordered frequent item list from Step 4. If different transactions have common prefixes (initial subsets of their itemsets), those transactions share the prefix paths, which leads to a compressed representation of the transaction database.

Step 6: Final FP-tree Structure:

The final FP-tree is shown on the right side of the flow chart. The nodes "F", "G", "A", "M", and "P" represent the respective items with their frequency counters. The "F" node, as the most frequent, starts the first branch. Following the transaction order, the "G" node comes next, and so on, with branches splitting off for different transactions. When items are repeated in different transactions, their respective nodes share branches, and their counters are incremented accordingly. "M:1" and "P:1" indicate the presence of items "M" and "P" in one additional transaction each. The flow chart is a visualization of the process of building an FP-tree from transaction data for frequent itemset mining. The FP-tree is a compact structure that allows for efficient mining of frequent item sets without generating candidate sets, making it faster and less memory-intensive than algorithms like Apriori.

Association Rule:

Frequent Itemset: This part of the chart shows all the item sets that meet a minimum support threshold. Support is the proportion of transactions in the database that contain the itemset. For example, itemset {A} appears in 50% of all transactions. Rules Generated from item sets: This part of the chart lists the association rules derived from the frequent item sets. An association rule is an implication of the form X => Y, where X and Y are disjoint item sets.

Confidence: The confidence of a rule is a measure of how often items in Y appear in transactions that contain X. It's the probability P(Y|X), the likelihood of Y given X. For example, the rule A => B has a confidence of 0.6, which means that in 60% of the transactions containing A, B also appears.

Support: The support of a rule is defined as the proportion of transactions that contain both X and Y to the total number of transactions. It indicates how frequently the itemset appears in the dataset. For instance, the rule A => B has a support of 0.3, meaning that both A and B appear together in 30% of all transactions.

Lift: Lift is a measure of the strength of an association over the randomness. It's defined as the ratio of the observed support to that expected if X and Y were independent. A lift value greater than 1 indicates that the items are more likely to be bought together than randomly. For example, the rule B => C has a lift of 1.68, suggesting a positive association.

The chart below would typically be accompanied by a visualization, such as a network graph, with nodes representing items and edges representing the association rules, often with arrowheads pointing from the antecedent item(s) to the consequent item(s). The thickness of the edges might correspond to the confidence or support of the rules. In practice, association rule mining is conducted using specialized data mining tools that can handle large datasets and complex calculations.

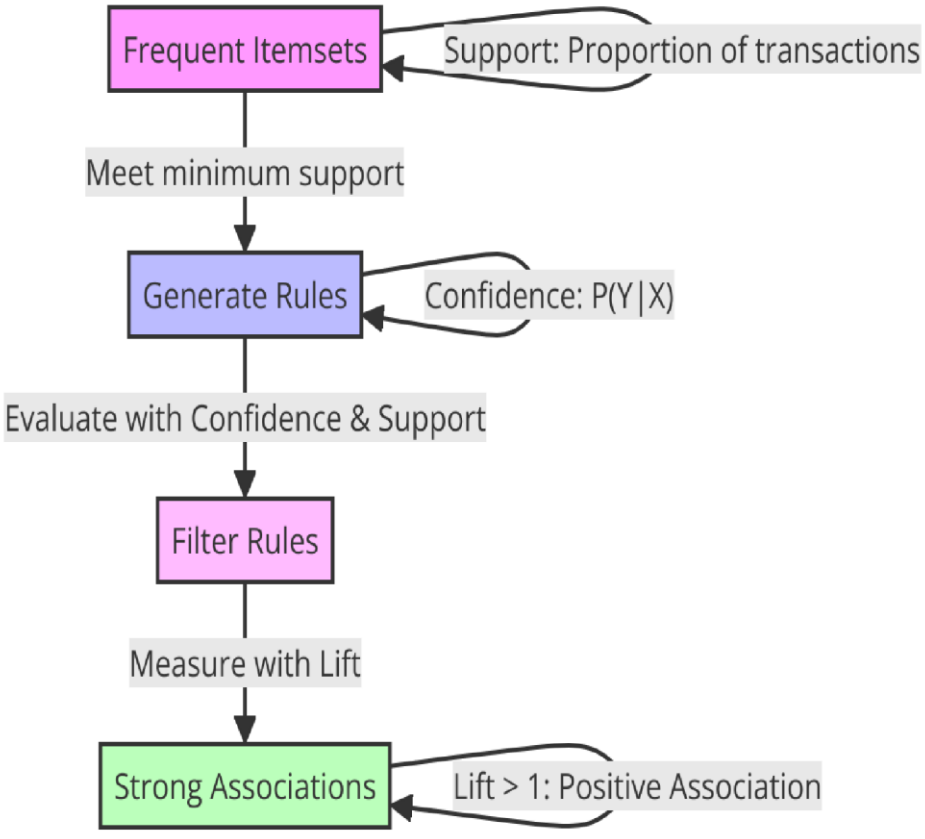


Figure Association Rule Flow Chart

Time Series Construction for Trend Analysis

With the target products identified, the dataset is filtered to include only the transactions that contain these products. Subsequently, the purchase dates associated with these transactions are converted into a datetime format, which is essential for accurate time series analysis. The conversion ensures that dates are recognized by the analytical tools in a consistent and computationally tractable manner. Following the date conversion, the dataset is further processed to extract the year and month from each purchase date. This extraction serves to consolidate the sales data into monthly periods, which is a common temporal resolution that balances detail with manageability in time series analysis.The pre-processed data is then aggregated to calculate the total revenue generated by each of the selected products on a month-by-month basis. This step condenses the transaction-level data into a higherlevel overview of sales performance, making it possible to observe and compare trends over time.

Trend Analysis and Visualization

Finally, the aggregated sales data is visualized using a time series plot that displays the revenue trends of the selected products over the observed period. Each product is represented by a distinct line on the graph, often with markers at each data point to enhance readability. The axes are labeled to reflect the temporal scale (Year-Month) and the revenue metric, while a legend is provided to identify the product lines. The visualization provides a clear and immediate means of comparing the sales trajectories of the highly associated products, offering insights into patterns such as seasonality, growth trends, or anomalies. The trend lines allow for the assessment of the relative performance of the products and the dynamics of their sales over time. The flowchart displays the whole process in a very clear way.

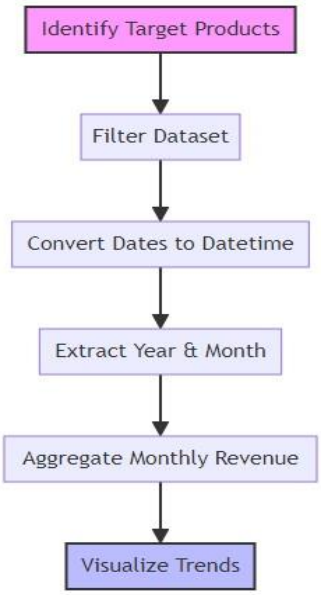


Figure Trend analysis flow diagram

5 Data Analysis

Data Loading

The Data of the csv file is loaded from the directory where the file was after the series of vital Python libraries are carried in. The 'pandas' library is chosen for its vigorous data structuring and analysis capabilities, 'numpy' for handling complex numerical tasks, while 'matplotlib' and 'seaborn' are selected for their advanced visualization tools. Additionally, 're' is used for manipulating strings with regular expressions, and 'networkx' is employed for its ability to analyse complex networks.

Once the environment is set up with these tools, attention is turned to the dataset itself. In the second step, a CSV file, aptly titled 'Market Basket Analysis(MBA).csv', is ingested into the workspace. This file is read into a DataFrame, the quintessential data structure in pandas, with special attention to encoding to ensure any unique or accented characters are correctly interpreted.

To confirm a effective load and to get a sense of the data's makeup, the third step involves showcasing the first ten records of the DataFrame. This glimpse reveals a structured table populated with several columns: product names, categories, unique transaction identifiers, quantity sold, and the resulting revenue from these transactions. This brief exploration serves as a foundation upon which further, more intricate analysis can be constructed, possibly leading to insights on purchasing patterns and customer behaviour.

Data type and Statistical analysis

Initially, a comprehensive overview of the dataset is obtained using the df.info() function. This reveals that the dataset contains 92,250 entries across five columns. Each column is densely populated with non-null values, indicating a dataset without missing values. The columns encompass details such as product names, categories, transaction identifiers, quantities sold, and product revenues. Notably, while the 'Quantity' column is stored as integers, the rest are recognized as objects, a datatype often used for representing text in pandas. Recognizing the need for numerical manipulation, the analyst then addresses a common preprocessing step: the conversion of the 'Product Revenue' column to a float datatype. This transformation is necessary because the column contains non-numeric characters, specifically currency symbols, that could impede mathematical operations. A regular expression is employed to strip these symbols, ensuring the data is cast correctly to floats. With the data now properly formatted, descriptive statistics are generated using the df.describe() function. This step is crucial as it provides insights into the central tendencies and dispersion of the numerical fields. For

instance, it indicates that, on average, the quantity of products sold per transaction is slightly above one, and the mean revenue per product stands at around 190 units of currency. Finally, the dimensions of the DataFrame are confirmed through the df.shape attribute. The output confirms the presence of 92,250 rows and 6 columns, solidifying the foundational understanding of the dataset's scale and scope.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92250 entries, 0 to 92249
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Purchase Date     92250 non-null  object
 1   Product           92250 non-null  object
 2   Product Category  92250 non-null  object
 3   Transaction       92250 non-null  object
 4   Quantity          92250 non-null  int64
 5   Product Revenue   92250 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 4.2+ MB
```

Figure Data type

	Quantity	Product Revenue
count	92250.000000	92250.000000
mean	1.164564	190.383006
std	0.676474	297.407428
min	1.000000	0.000000
25%	1.000000	44.990000
50%	1.000000	116.990000
75%	1.000000	219.980000
max	47.000000	14739.200000

Figure Data Statistics

Exploratory Data Analysis(EDA)

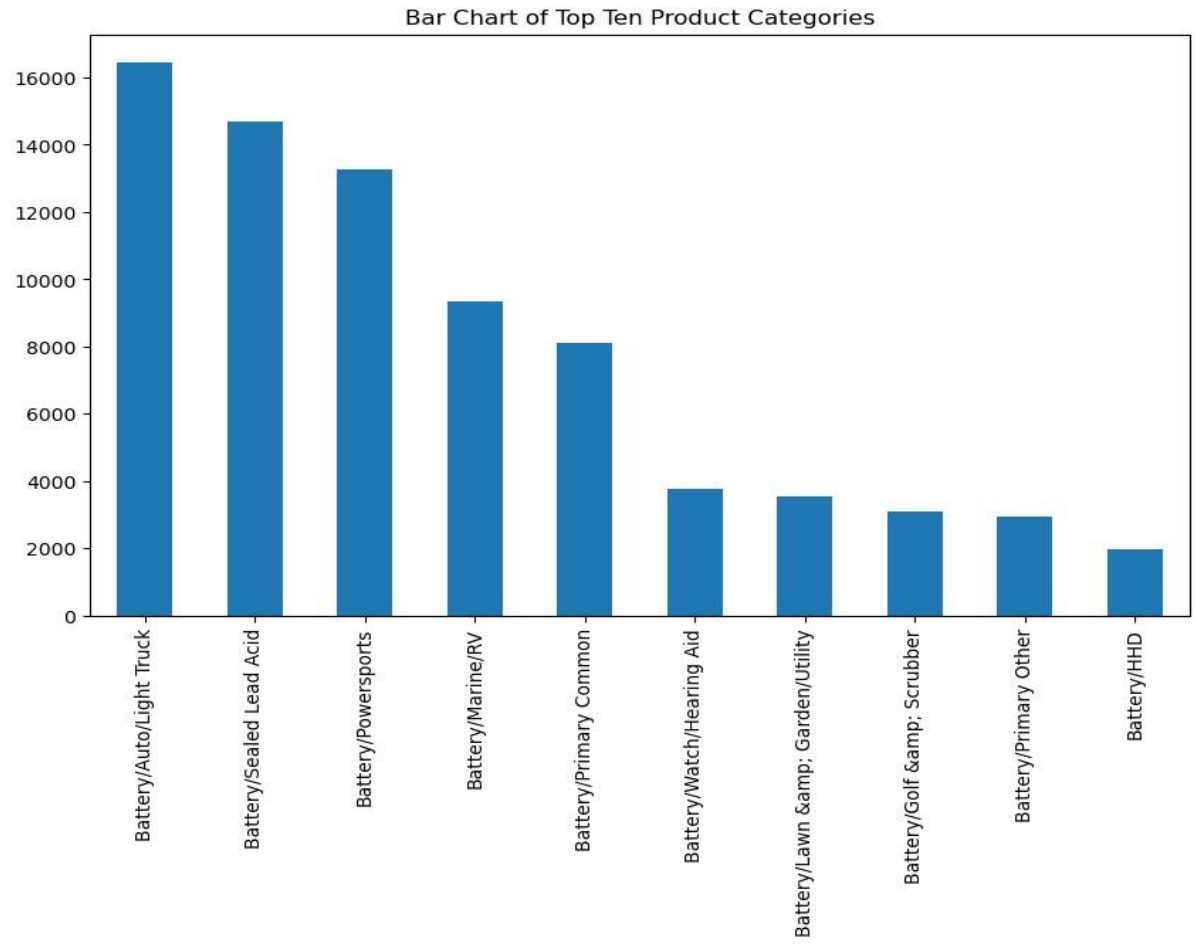


Figure Bar Chart of Product Category

The chart above provided is a bar chart titled "Bar Chart of Top Ten Product Categories." It displays the frequency or count of items in each product category. The categories appear to be related to various types of batteries and the products or applications they are associated with. Here's a breakdown of the categories from the left to the right:

- Battery/AVL/Light Truck - This category has the highest count, reaching just above 16,000.
- Battery/Sealed Lead Acid - The second category is slightly less than the first, with a count around 15,000.
- Battery/Powersports - The third bar shows a count just above 12,000.
- Battery/Marine/RV - The fourth bar is shorter, indicating a count of about 10,000.
- Battery/Primary/Common - This category has a count of just over 8,000.
- Battery/Watch/Hearing Aid - The sixth bar is slightly lower than the fifth, around 7,000.
- Battery/Lawn & Garden/Utility - This category has a count of approximately 6,000.
- Battery/Golf & Scrubber - The eighth bar shows a count just below 6,000.
- Battery/Primary/Other - The ninth category has a count just above 4,000.
- Battery/HHD - The last category on the chart has the lowest count, around 2,000.

The x-axis labels the product categories, while the y-axis represents the count, with increments of 2,000 up to 16,000. Each bar represents the count of items within a specific product category. The chart is a useful way to quickly compare the counts of items across these top ten battery-related product categories.

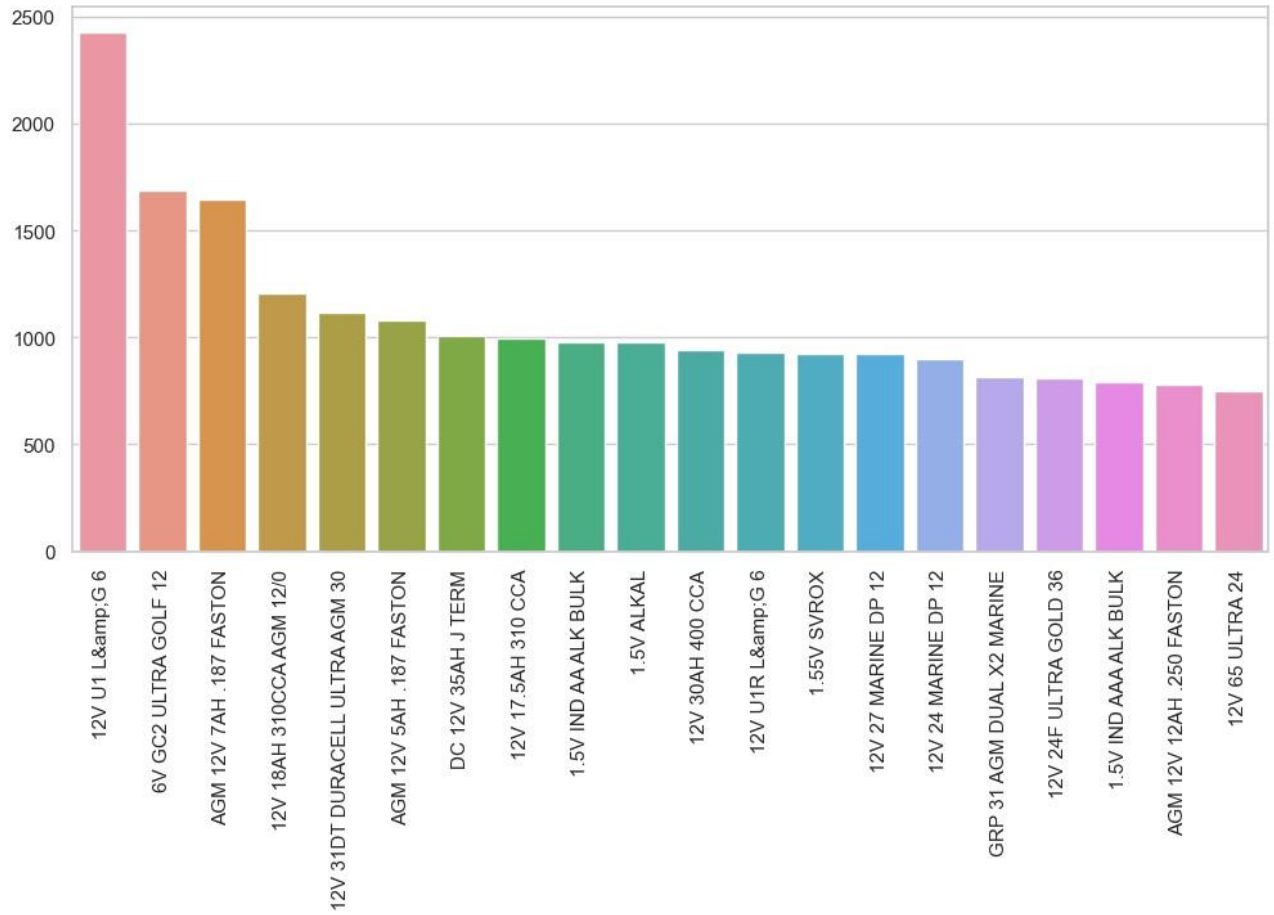


Figure The Bar graph of top Products

The bars are organized in descending order from the highest count to the lowest. The x-axis represents the count of the batteries, with a scale that seems to increase by increments of 500 up to a maximum of 2500, while the y-axis lists the battery categories with specific details such as voltage, capacity (in Ampere-hour)

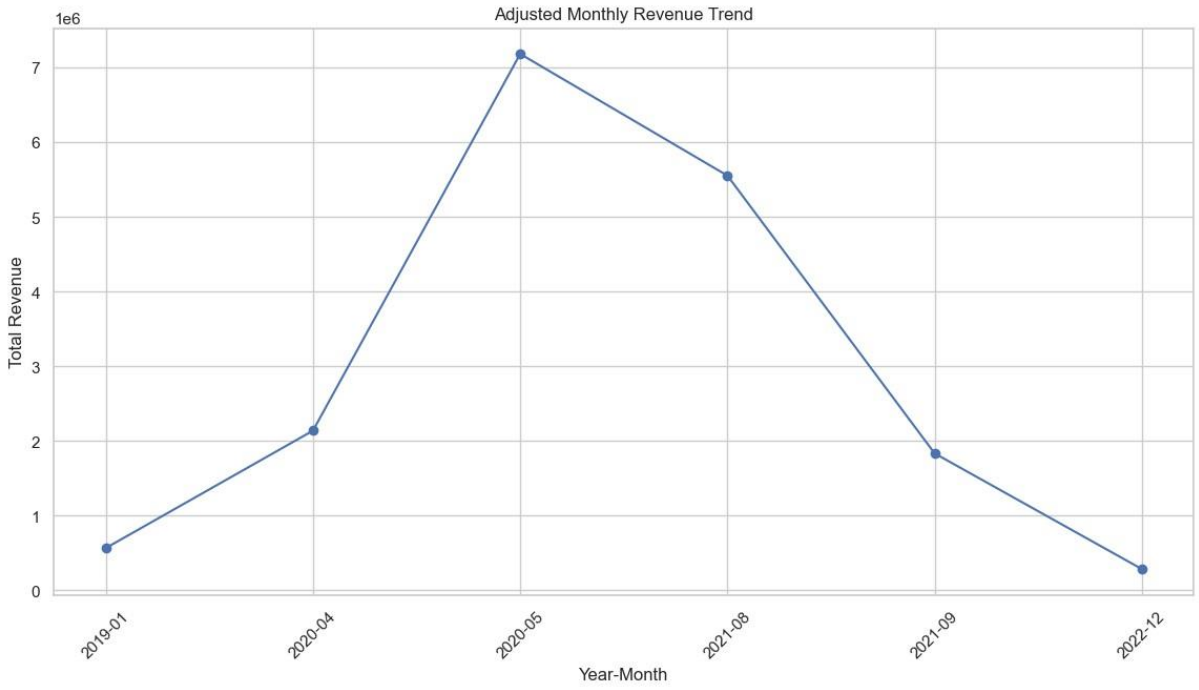


FIGURE Line Graph of monthly revenue

The line graph with the title "Adjusted Monthly Revenue Trend," which tracks the total revenue over time from an unspecified source. The x-axis represents time, labeled with dates in a year-month format (e.g., "2019-01" for January 2019), suggesting a monthly interval between data points. The yaxis represents total revenue, scaled to 1e6 (which denotes 1 million when using scientific notation), indicating that the values are in the millions.

The graph displays a trend where the revenue starts at a lower point, increases to a peak, and then declines. Specifically, the line begins with a point that suggests revenue was just above 1 million, rises steadily to a peak of nearly 7 million around the midpoint of the graph, and then falls to just above 1 million again by the end of the period displayed. The highest point is at "2020-05," indicating the peak revenue was achieved in May 2020.

This graph might be used to demonstrate seasonal trends, the impact of a specific event or campaign, or perhaps the lifecycle of a product or service in the market. The precise reasons for the rise and fall in revenue would require additional context related to the data.

Data Cleaning

The first action undertaken is the removal of duplicate records—a common yet pivotal step in data preprocessing—accomplished using the df.drop_duplicates() method. This step is fundamental to prevent any bias that replicated entries might introduce into the analysis.

Following this, the analyst probes the dataset for any missing entries across all columns, employing the df.isnull().sum() method. The output reassuringly indicates a complete dataset, devoid of any null values, thus confirming the robustness of the dataset in terms of completeness. Moving on, the analyst targets specific data anomalies by eliminating rows where the Transaction column contains placeholder text <transaction id>. This refined operation is a testament to the detail-oriented approach, ensuring that only meaningful data persists. To further strengthen the data's integrity, an additional check for duplicates, using df.duplicated()—a measure that underscores the thoroughness of the process, as it reaffirms the absence of any remaining duplicates. Lastly, the dataset is filtered to ensure that it reflects only positive transactional activities, by keeping rows where the Quantity is greater than or equal to zero. This step is significant as it potentially filters out return transactions or data entry errors, thus purifying the dataset for analysis that will likely focus on sales data.

Each of these conscientiously executed steps collectively forms a comprehensive data cleaning regimen. By methodically ensuring that the dataset is free of redundancies, voids, and inaccuracies, the analyst lays a solid groundwork. Such diligence is indispensable, as it facilitates the extraction of reliable insights and supports the credibility of any future statistical modelling, data-driven decisionmaking derived from this cleansed dataset.

Apply k-means Clustering Cluster

Aggregating Transaction Data

The analysis begins by consolidating transactional information. Each transaction in the dataset might consist of multiple entries, each representing a different product or a repeat purchase. The aggregation process sums up the Quantity and Product Revenue for each unique transaction, transforming the dataset into a simpler form where each row represents a single transaction with its total quantity and revenue. This step is crucial for reducing complexity and preparing the data for meaningful analysis.

Standardizing Features for Clustering

After aggregation, the next critical step is standardizing the features, specifically Quantity and Product Revenue. Standardization is a preprocessing technique used to ensure that each feature contributes equally to the analysis, especially in algorithms like K-Means that compute distances between data points. Without standardization, features with larger scales dominate the distance calculations, leading

to biased clusters. By transforming the data such that each feature has a mean of 0 and a standard deviation of 1, we ensure a fair comparison and contribution of each feature to the clustering process.

Optimal Number of Clusters: Elbow Method

Determining the optimal number of clusters is a pivotal step in cluster analysis. The Elbow Method is a heuristic approach used to find a balance between the number of clusters and the sum of distances of points from their respective cluster centroids (within-cluster sum of squares, WCSS). As the number of clusters increases, WCSS tends to decrease; the goal is to find the point where the reduction in WCSS starts to diminish, indicating that adding more clusters does not significantly improve the fit. This "elbow" point is subjectively chosen from the plot, marking the optimal number of clusters for the K-Means algorithm.

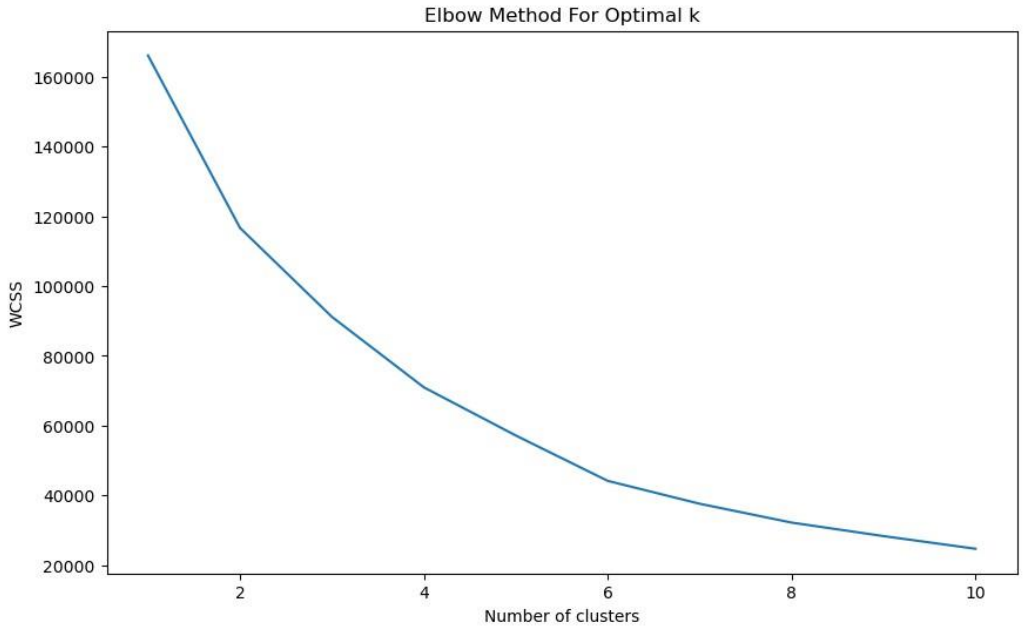


Figure Elbow Method

The above provided is a visual representation of the Elbow Method used to determine the optimal number of clusters for K-Means clustering. Here's what the chart illustrates:

The x-axis represents the number of clusters tested in the K-Means algorithm, ranging from 1 to 10.

The y-axis shows the within-cluster sum of squares (WCSS), which is a measure of the total variance within each cluster. The WCSS decreases as the number of clusters increases because the data points are closer to the centroids they are assigned to.

In the Elbow Method, one looks for the point where the decrease in WCSS begins to slow down, creating an "elbow" shape in the graph. This is considered to be the point beyond which adding more clusters doesn’t provide a significant improvement in the variance explained by the clusters.

In the provided chart, the curve is smooth, and there's no clear "elbow" where the WCSS stops decreasing significantly. However, there seems to be a slight bend around the cluster count of 3 or 4. This slight bend could be interpreted as the elbow, suggesting that increasing the number of clusters beyond this point might not yield much better clustering in terms of WCSS

Applying K-Means Clustering

With the optimal number of clusters identified, the K-Means algorithm is applied to the standardized data. K-Means is an iterative algorithm that partitions the data into k mutually exclusive clusters, minimizing the variance within each cluster. The algorithm assigns each data point to the nearest cluster centroid and then recomputes the centroids based on the current cluster memberships, repeating this process until the cluster assignments stabilize. The initialization parameters and the random state ensure consistent and reliable clustering results.

Assigning Cluster Labels and Merging

Each transaction is assigned a cluster label based on its proximity to the cluster centroids. These labels are then merged back into the original dataset, enriching it with new insights. This step is crucial for subsequent analysis, as it allows for the examination of transactions within the context of their cluster memberships.

Examining Clustered Products

The enriched dataset enables an analysis of the products within each cluster. By grouping the transactions by cluster and listing unique products, we gain insights into the product composition of each cluster. This step is valuable for understanding the characteristics that define each cluster, such as common products or product categories that might indicate similar purchasing behaviors or preferences.

Merging and Grouping for Detailed Analysis

If not already done, the cluster labels are ensured to be merged with the original transaction data. The dataset is then grouped by Cluster and Product, and the occurrences of each product within each

cluster are counted. This detailed grouping and counting provide a foundation for deeper analysis of product distribution across clusters.

Identifying and Displaying Top Products

The final step involves sorting these counts within each cluster to identify the top products. This is achieved by selecting the products with the highest counts within each cluster, highlighting the most prevalent products. Displaying these top products offers valuable insights into the dominant purchasing patterns within each cluster, revealing potential market segments or customer preferences.

By elaborating on each step, we gain a comprehensive understanding of the methodology, from data preparation through clustering to the detailed analysis of clustered transactions. This process not only uncovers hidden patterns in transaction data but also provides a structured approach to market segmentation and customer behaviour analysis.

	Cluster	Product	counts
0	0	12V U1 L&G 6	2308
1	0	AGM 12V 7AH .187 FASTON	1443
2	0	12V 18AH 310CCA AGM 12/0	1136
3	0	AGM 12V 5AH .187 FASTON	998
4	0	DC 12V 35AH J TERM	901
5	1	6V GC2 ULTRA GOLF 12	29
6	1	AGM 12V 7AH .187 FASTON	19
7	1	AGM 12V 12AH .250 FASTON	17
8	1	8V GC8 ULTRA 12	16
9	1	12V 31DT DURACELL ULTRA AGM 30	15
10	2	6V GC2 ULTRA GOLF 12	759
11	2	1.5V IND AA ALK BULK	457
12	2	1.5V IND AAA ALK BULK	438
13	2	8V GC8 ULTRA 12	399
14	2	1.5V ALKAL	360

Figure of Clustered Products

The figure chart is explained below:

Cluster: This column represents the different groups that the transactions have been divided into, based on the clustering algorithm. The numbers (0, 1, 2, etc.) are identifiers for the clusters. Typically, transactions in the same cluster should exhibit similar characteristics, such as similar quantities purchased, or revenue generated.

Product: It lists the unique products that have been purchased within each cluster. The names seem to refer to various battery types, which suggests that the data pertains to a company or a market segment dealing with battery sales. The products are identified by specific characteristics such as voltage (12V, 6V, 8V), type (AGM, DC, IND AAA), and possibly model or size (e.g., ULTRA GOLF 12, AALK BULK).

Counts: This column indicates the number of times each product appears within the corresponding cluster. For instance, '12V U1 L&G 6' appears 2308 times in Cluster 0, suggesting it is a common product in that cluster.

Cluster 0 is the most populated with the '12V U1 L&G 6' product having the highest count, indicating that this product is perhaps the most popular or most frequently sold item in this cluster.

Cluster 1 has much lower product counts compared to Cluster 0, which could imply that the transactions in this cluster are less frequent or that there's a wider variety of products being purchased in smaller quantities.

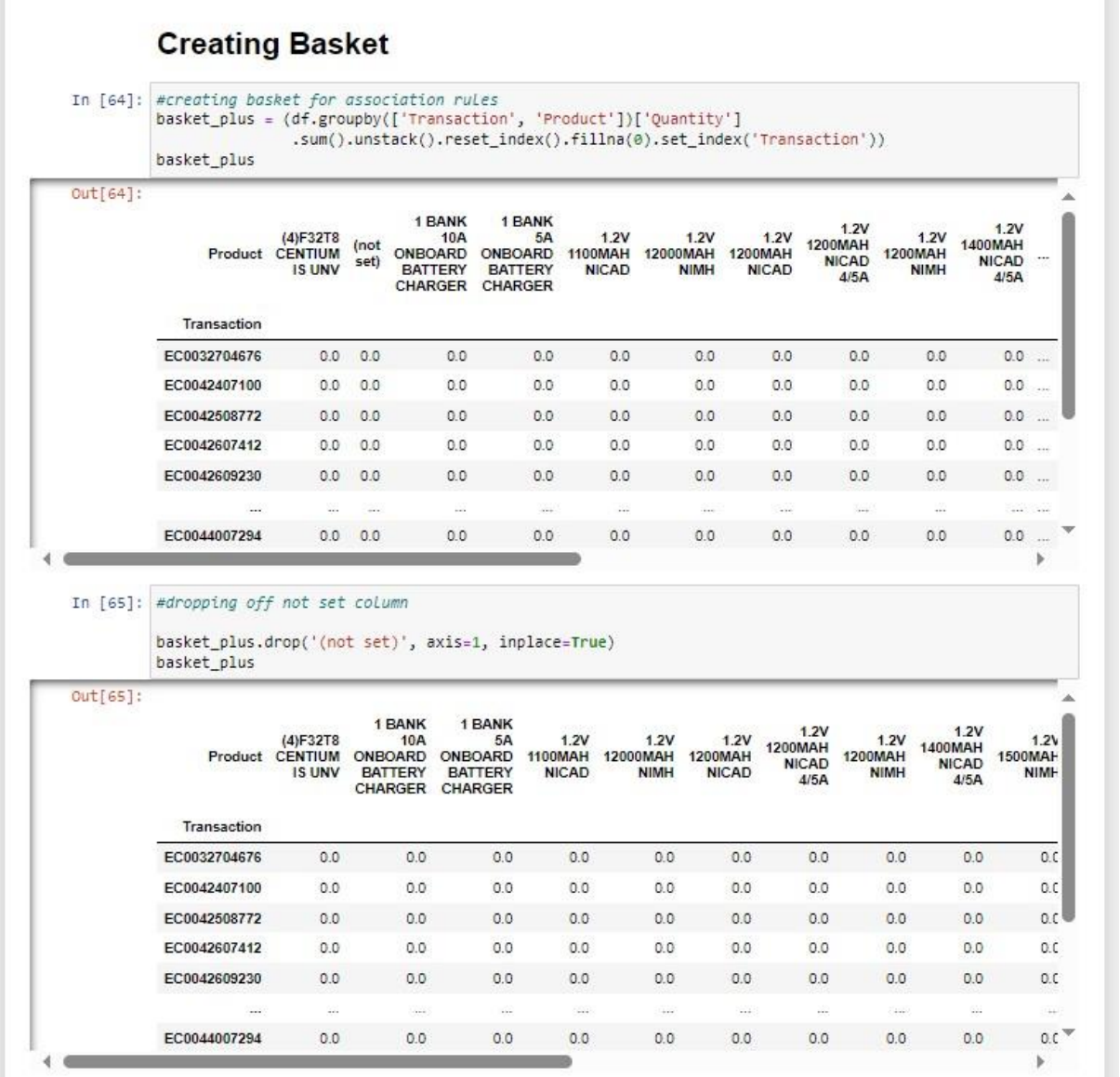
Cluster 2 has even lower counts for its top products compared to Cluster 1, which could indicate even less frequent purchases or a greater diversity in the product types within this cluster indicating the products that are bought together. For further analysis we will go for association rules.

Basket Creation/Feature Selection

In this segment of the analysis, the dataset undergoes a transformation to match a customer's shopping basket. The data is first organized by grouping according to each transaction, identified by 'Transaction', and the specific products purchased, referred to as 'Product'. Within these groups, the 'Quantity' field is aggregated to reflect the total number of each item bought within a single transaction. Following the combination, the data is unstacked, a process that pivots the product entries from a stacked (hierarchical) index to a more readable, columnar format. This restructuring is akin to laying out the contents of a shopping basket for each transaction, making it possible to observe the quantities of individual items.

The final touch in this transformation is adjusting the DataFrame's index to use the 'Transaction'. This adjustment allows for a clear visualization of each customer's transaction as a discrete entity, akin to a shopping basket at checkout. Here, the quantities of different products are laid out transaction-wise, where a '0' indicates that the particular item was not part of the transaction, and any other number reflects the quantity of that item purchased. For instance, a '1'0 would signify that the customer bought ten units of the item.

This reformed DataFrame offers a detailed look into customer purchasing patterns, revealing how many of each item were bought in every transaction, which is instrumental for understanding customer behavior, such as identifying commonly purchased items or combinations of products that are frequently bought together.



Creating Basket

In [64]:
```
#creating basket for association rules
basket_plus = (df.groupby(['Transaction', 'Product'])['Quantity']
               .sum().unstack().reset_index().fillna(0).set_index('Transaction'))
basket_plus
```

Out[64]:

Product	(4)F32T8 CENTIUM IS UNV	(not set)	1 BANK 10A ONBOARD BATTERY CHARGER	1 BANK 5A ONBOARD BATTERY CHARGER	1.2V 1100MAH NICAD	1.2V 12000MAH NIMH	1.2V 1200MAH NICAD	1.2V 1200MAH NICAD 4/5A	1.2V 1200MAH NIMH	1.2V 1400MAH NICAD 4/5A	...
Transaction											
EC0032704676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
EC0042407100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
EC0042508772	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
EC0042607412	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
EC0042609230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
EC0044007294	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

In [65]:
```
#dropping off not set column

basket_plus.drop('(not set)', axis=1, inplace=True)
basket_plus
```

Out[65]:

Product	(4)F32T8 CENTIUM IS UNV	1 BANK 10A ONBOARD BATTERY CHARGER	1 BANK 5A ONBOARD BATTERY CHARGER	1.2V 1100MAH NICAD	1.2V 12000MAH NIMH	1.2V 1200MAH NICAD	1.2V 1200MAH NICAD 4/5A	1.2V 1200MAH NIMH	1.2V 1400MAH NICAD 4/5A	1.2V 1500MAH NIMH
Transaction										
EC0032704676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EC0042407100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EC0042508772	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EC0042607412	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EC0042609230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
EC0044007294	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure Basket Creation

The above figure portrays the DataFrame df is being manipulated to group the data by 'Transaction' and 'Product'. For each group, the 'Quantity' is aggregated using the sum() function. The resulting DataFrame is then reset to ensure that 'Transaction' is no longer the index of the DataFrame but a regular column, which is then filled with zeroes in place of any NaN values that may arise from the grouping and aggregation process. Finally, the 'Transaction' column is set as the index again, resulting in a matrix where each row represents a unique transaction and each column represents a different product. The cells within the matrix contain the quantity of the respective product that was bought in

each transaction. This sort of matrix is typically used in Market Basket Analysis(MBA) to determine which products are often purchased together. The it follows through a post-processing cleanup where a column named '(not set)' is dropped from the 'basket_plus' DataFrame. This is done using the drop method with the arguments ('not set', axis=1, inplace=True), which specifies that a column (axis=1) named 'not set' should be removed from the DataFrame directly (without needing to assign the result to a new DataFrame), as indicated by inplace=True. The resulting output is a cleaner 'basket_plus' DataFrame without the unwanted '(not set)' column, ready for further analysis.

Encode

While preparing the dataset, a function called encode_units which is designed to encode the quantities into a binary format: if the quantity is less than or equal to 0, it returns 0; if the quantity is greater than 0, it returns 1. This encoding is applied across the basket_plus DataFrame using the applymap function, which applies a given function to each element of the DataFrame. The resulting DataFrame, basket_encode_plus, contains binary values where each cell indicates the presence (1) or absence (0) of a product in each transaction. This binary transformation is a standard step in preparing data for Market Basket Analysis(MBA) since the analysis focuses on the presence rather than the quantity of items in transactions. Then the dataset is refined further by focusing on transactions where more than one item has been purchased. This is achieved by summing up the encoded values across each row (using sum(axis=1)) and filtering the DataFrame to include only those transactions with sums greater than or equal to 2. The resulting basket_filter_plus DataFrame is a subset of basket_encode_plus, where only transactions with two or more items are retained.Finally, the filtered DataFrame, which now has 6,321 rows, indicating that many transactions with only one item have been excluded from this analysis. This approach ensures that the Market Basket Analysis(MBA) will consider only transactions with multiple items, which can provide insights into the relationships and patterns of products commonly purchased together.

Implementation Frequent Pattern Growth

FP-Growth algorithm has been adeptly employed to unearth frequent itemsets within the basket_filter_plus DataFrame. This DataFrame, which represents a filtered transactional dataset, serves as fertile ground for the algorithm to operate on. The FP-Growth algorithm, renowned for its efficiency in handling large datasets, is set to identify itemsets that appear in at least 3% of transactions, a parameter articulated by the minimum support threshold of 0.03. By setting

use_colnames=True, the algorithm is configured to output actual product names, facilitating an intuitive interpretation of the results.

The execution of the algorithm culminates in the generation of the frequent_itemsets DataFrame. This DataFrame catalogues each itemset with an accompanying support value, quantifying the frequency of occurrence across all transactions. To enrich the analysis, a 'length' column is ingeniously integrated into the frequent_itemsets DataFrame, denoting the number of items in each set through the application of a lambda function. This additional detail paints a clearer picture of the itemsets' complexity, portraying the scope of multi-item purchases. Upon printing, the frequent itemsets and their lengths are laid bare, revealing not only the support but also the relational intricacy within each itemset. Such insights are paramount for appreciating the nuanced purchasing patterns that might otherwise be obscured. Prior to the application of FP-Growth, the notebook logs the installation of the mlxtend package, a vital toolkit for pattern mining. The seamless installation attests to the readiness of the environment, with all dependencies already in place. A cautionary note from mlxtend warns of the impending deprecation for non-boolean DataFrame inputs, advising a shift to binary representations to enhance computational efficiency. Here,a sample of the frequent_itemsets DataFrame. An example entry discloses an itemset with a single item—(1.5V ALKALINE AA CONTRACTOR 24PK)—which commands a significant presence in the transactional dataset, featuring in 8.9719% of the transactions. This strategic application of the FP-Growth algorithm, favored for its adept handling of large datasets where the conventional Apriori might falter, leverages a divide-andconquer methodology. It constructs an FP-tree, a compact structure encoding transactions, thereby circumventing the exhaustive candidate generation phase. Such an approach not only streamlines the discovery of frequent patterns but does so with a remarkable alacrity that is particularly beneficial when sifting through voluminous datasets.

While running the codes, the Apriori took time while FP growth algorithm sorted out very fast. Though both portrays same results so for further analysis FP growths datasets will be chosen.

Implementation Apriori

The apriori function from the mlxtend.frequent_patterns module is called to identify frequent itemsets within the basket_filter_plus DataFrame, which likely contains binary transaction data.The min_support parameter is set to 0.03, indicating that the algorithm will only consider itemsets that appear in at least 3% of the transaction dataset, ‘use_colnames’ is set to True, which means the resulting DataFrame will display the actual item names instead of column indices. The resulting frequent_itemsets_plus DataFrame is then sorted by the 'support' column in descending order (ascending=False), bringing the most common itemsets to the top. Then the DataFrame index is reset

using reset_index(drop=True), which removes the old index and replaces it with an integer sequence. A new column named 'length' is added to the frequent_itemsets_plus DataFrame, which shows the number of items in each itemset. This is done by applying a lambda function to each itemset that calculates its length.

The output shows the frequent_itemsets_plus DataFrame with columns 'support', 'itemsets', and 'length'. 'Support' indicates how frequently the itemset occurs in the data as a proportion of the total transactions. 'Itemsets' are the actual items appearing together frequently in transactions. 'Length' indicates the number of items in each itemset.

The warning advises to use a DataFrame with boolean types (0s and 1s) for item presence to ensure compatibility and optimal performance.

In the sample output, we see various itemsets with their respective support values and lengths. For example, the first itemset includes (1.5V IND AA ALK BULK) and has a support of 0.115013, indicating it appears in about 11.5013% of the transactions, and it consists of just one item, as denoted by the length of 1. The Apriori algorithm is a fundamental method for mining frequent itemsets and is often used prior to generating association rules. It operates by generating candidate itemsets and then pruning those that do not meet the minimum support threshold, iteratively expanding the itemsets if their support is above the given threshold.

Association Rules

Since the itemsets from fp-growth gives faster outcome we will chose this to go forward this. Association Between Frequently Bought Items & finding the best products the association_rules function from the mlxtend.frequent_patterns module plays a crucial role in extracting meaningful patterns from a dataset of frequent itemsets. This analysis probes the intricate web of transactions to identify which items, known as antecedents, are likely to influence the purchase of other items, called consequents. The lift metric, a key parameter in this function, evaluates the strength of these relationships beyond mere chance, with a minimum threshold set to ensure only significant correlations are considered. The resulting association rules are then meticulously sorted by lift in descending order, providing a clear hierarchy of itemset associations.

The DataFrame, a structured output from this process, summarizes the essence of these rules through several descriptive metrics. The antecedent and consequent supports provide a description of how common these items are in transactions, while the combined support offers a joint occurrence frequency. Confidence delves deeper, calculating the probability of encountering the consequents when the antecedents are present. Lift goes a step further, comparing the observed frequency of item co-occurrence against the frequency expected under independence, effectively highlighting the

potency of the association. Leverage and conviction are additional metrics that dissect the dependency and reliability of these associations, while Zhang's metric offers an alternative perspective on their significance.

A narrative woven from these statistics emerges when interpreting the rules. For instance, one might uncover that purchasing (1.5V IND AA ALK BULK) significantly increases the likelihood of (1.5V IND AAA ALK BULK) being in the same basket—a relationship underscored by a lift value suggesting a 6.69fold increase over independent purchasing probability. Such insights are instrumental, revealing not just concurrent buying trends but the magnitude of their connection. This analytical vista into consumer behavior is invaluable, offering a compass for tactical decisions in marketing strategies, product placements, and inventory optimization, all tailored to foster symbiotic relationships between consumer desire and business offerings.

3D Scatter Plot

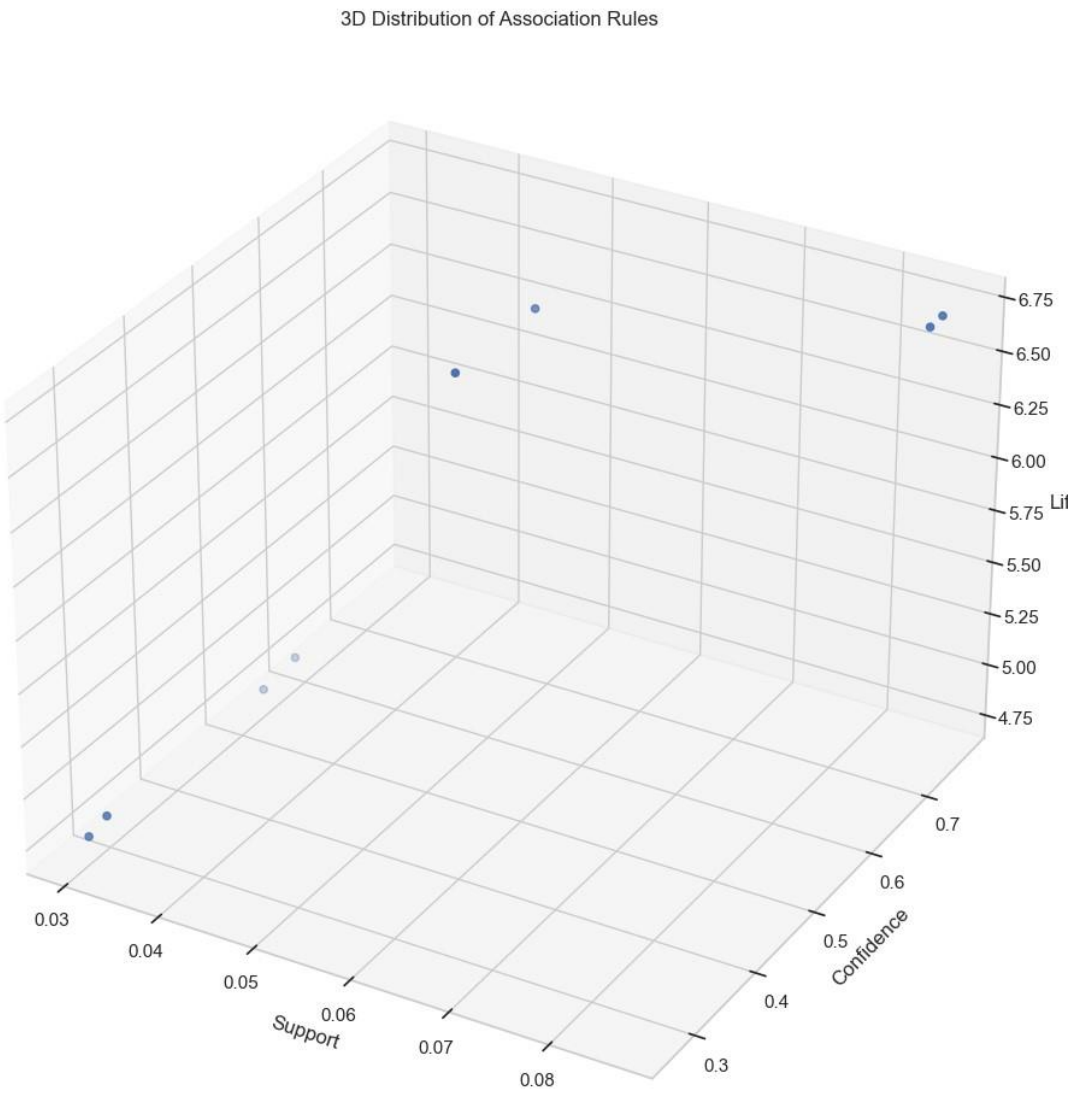


Figure Scatter plot

Support (X-axis): The support values indicate the frequency with which the itemsets appear in the dataset. For example, the itemset (1.5V IND AA ALK BULK) has a support of 0.115013, suggesting that this item appears in approximately 11.5% of all transactions. Similarly, the itemset (1.5V ALKAL) appears in roughly 9.9% of transactions, as indicated by its support value of 0.098719.

Confidence (Y-axis): Confidence values measure the likelihood of the consequent being purchased given that the antecedent is purchased. For instance, the confidence of 0.748281 for the rule where (1.5V IND AA ALK BULK) leads to (1.5V IND AAA ALK BULK) implies that there is a 74.8% chance of the latter being bought when the former is in the basket. Conversely, when (D ALKALINE BULK) is bought, there's a 53.4% chance of (1.5V IND AAA ALK BULK) being purchased as well, as evidenced by the confidence value of 0.533708.

Lift (Z-axis): Lift measures the strength of an association. A lift greater than 1 indicates that the antecedent and consequent appear together more often than expected if they were independent. For example, the rule involving (1.5V IND AA ALK BULK) and (1.5V IND AAA ALK BULK) has a lift of 6.699549, meaning that the presence of one in a transaction increases the likelihood of the other being present by approximately 6.7 times compared to random chance.

When plotted, each point (or 'beat') in the scatter plot is positioned according to these three values. The plot provides a visual means to rapidly identify which rules are most compelling. For instance, those with higher lift and confidence might be particularly interesting, as they suggest strong purchase patterns. The points associated with the highest lift and confidence in this dataset involve the itemsets with (1.5V IND AA ALK BULK) and (1.5V IND AAA ALK BULK), both having a lift over 6.69 and confidence levels above 74%, making them standout associations.

The leverage and conviction values provide additional context to these relationships, with higher leverage indicating a greater than expected frequency of the itemset occurrence, and higher conviction suggesting a stronger dependency of the consequent on the antecedent. Zhang's metric further quantifies the direction and strength of the rules, with values close to 1 indicating a strong positive association**.**

Network Graph

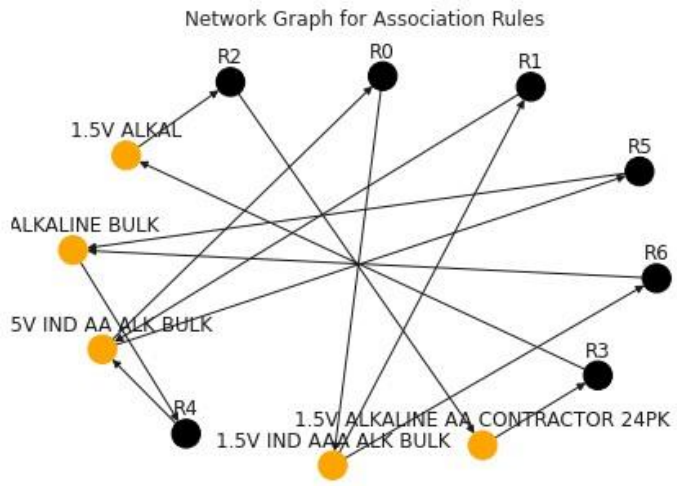


Figure Network Graph

Nodes (Circles): Each node represents an item or an itemset. The size of the nodes often correlates with the support or another metric (like frequency of occurrence) of the item—in this case, the items include "1.5V ALKAL," "ALKALINE BULK," "5V IND AA ALK BULK," and "1.5V ALKALINE AA CONTRACTOR 24PK."

Edges (Lines): The edges connect items that are found together in transactions. The thickness of an edge might represent the confidence or the lift of the association rule—thicker lines can indicate stronger associations.

Colours: The color of a node might denote a particular attribute. In some visualizations, color can represent different categories or even metrics such as lift—here, it seems that "1.5V ALKAL," "ALKALINE BULK," and "1.5V IND AAA ALK BULK" are one category (orange), and "1.5V ALKALINE AA CONTRACTOR 24PK" is another (black).

Labels: Nodes are often labeled with the item name, and in some cases, edges might be labeled with the rule or metric they represent. Here, nodes are labeled with the product names, and there are labels like "R0," "R1," up to "R6," which might represent individual rules derived from the association analysis. The network graph provides a visual summary of how different items are associated with one another, allowing for an intuitive understanding of the data that can reveal insights into consumer purchasing behavior. For instance, if "5V IND AA ALK BULK" has many connections, it might be a central item often bought with many other items. If "R0," "R1," etc., represent individual rules, their positioning and connections can indicate how different rules relate to the same items.

Interpretation of rules

Looking at the data we crunched from those shopping baskets, there's a standout pair of items that just seem to click with each other—those would be the '1.5V IND AA ALK BULK' and '1.5V IND AAA ALK BULK'. They've got what we call a 'lift' value that's through the roof at 6.699549, which is basically number cruncher talk for saying they're flying off the shelves together way more often than you'd expect by chance. The bigger this lift number is, the more likely you're not just seeing a fluke. Anything over a lift of 1, and it's like those items are best buddies.

Now, when you see these two battery packs together in 8.60% of all the carts rolling up to the checkout, that's not just small talk. We're talking 544 times in total, considering we've got 6321 transactions to look at. That's the 'support' value and how often these batteries are getting bought in tandem.

However, Confidence is all about which item starts .So if '1.5V IND AA ALK BULK' is often the first one in the basket, and '1.5V IND AAA ALK BULK' follows, then rule number 1 is what's happening on the Basket. And that's exactly what's going down here—the AA's are paving the way for the AAA's.

Behavioral analysis with time

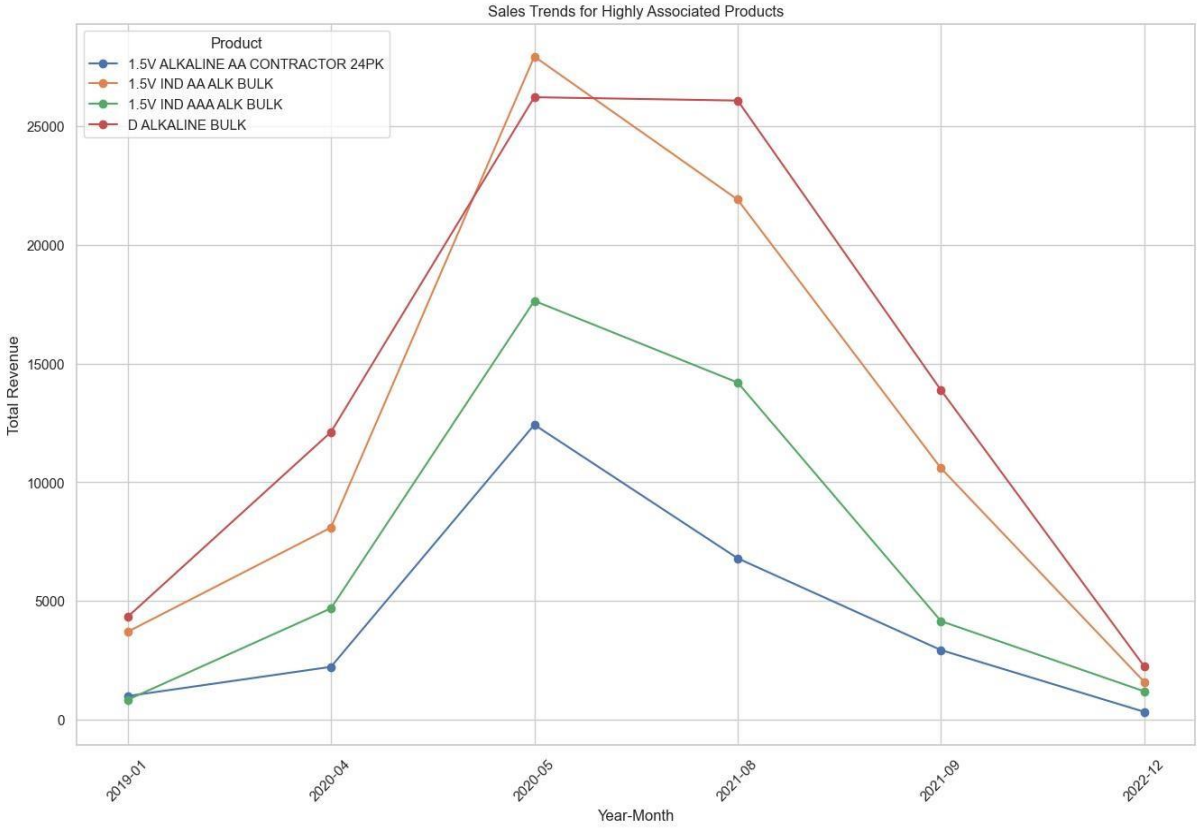


Figure sales trend

The graph provided illustrates sales trends for highly associated products over a period of time, specifically for a variety of batteries. The x-axis represents the time in a year-month format (YYYYMM), and the y-axis shows the total revenue generated by sales of these products.

From the line-graph , we can see that there is a significant increase in sales for all the products starting around the year-month 2021-04, reaching a peak at 2021-08. This could be interpreted as a result of the COVID-19 pandemic, which led to many people staying at home more than usual. During this time, the demand for batteries may have increased due to a higher usage of indoor electronic devices and home games that require batteries. This would be consistent with the behaviour of people seeking indoor entertainment options due to restrictions on outdoor activities and social gatherings.

Following the peak, there is a notable decline in sales for all the products by 2021-12. This downward trend might suggest that as the situation with COVID-19 improved, with vaccination drives or easing of restrictions, people started to spend more time outdoors and less time engaged in activities that

require batteries at home. Consequently, the demand for batteries decreased, leading to lower sales figures.

Overall, the graph could be reflecting consumer behaviour changes due to the pandemic, with an initial increase in sales as people adapted to home-based activities, followed by a decrease as they returned to more outdoor pursuits.

6 Data Driven Marketing Strategy

In the world of online shopping, making things easier and more rewarding for customers is key to success. By looking at what customers buy together, our website can make shopping smoother and maybe even save them some money. We found that certain items, like the '6V GC2 ULTRA GOLF 12', '1.5V IND AAA ALK BULK', '1.5V IND AA ALK BULK', '1.5V ALKAL', and '1.5V ALKALINE AA CONTRACTOR 24PK', often end up in shopping carts together. Let's use this info to make our customer's lives a bit easier. A demonstration through these product are given below in the next section.

Item Placements

Strategically, it's a savvy move to place the '1.5V IND AAA ALK BULK' alongside the '1.5V IND AA ALK BULK'—maybe on the same line or in another spot where they're neighbours. It’s about making it as convenient as possible for customers to find and pick up both items, almost without thinking. It's a bit like finding peanut butter right next to the jelly; they just naturally go together, and it simplifies the decision-making process for the shopper.

Products Bundling

Why not take it a step further and couple '1.5V ALKAL' with '6V GC2 ULTRA GOLF 12' in a tempting combo deal? Offer the pair at a price that's just a tad lighter on the wallet than buying them separately. This could be the nudge that shoppers need to grab the duo without a second thought, boosting sales and, ultimately, revenue. After all, who doesn’t love getting a little more for a little less?

Customer Recommendation and Discounts

Imagine this: a shopper is at the cashier, ready to buy the '1.5V IND AA ALK BULK', and right there, within easy reach, is the '1.5V ALKAL'. The cashier rings up the AA batteries and casually suggests the AA pack, mentioning the discount that comes with it. It's spontaneous, it's convenient, and it feels like getting an insider tip. This small interaction not only could increase sales but also enhances the customer experience with a personalized touch.

In addition to these strategies, leveraging the power of data analytics to offer personalized promotions could be a game-changer. For instance, analysing purchase histories to anticipate when customers might need new batteries and sending them a timely offer or discount. This approach ensures that the

deal lands just when the customer is most receptive. It's about staying one step ahead and showing customers that their needs are anticipated and met with precision and care.

7 Results Discussion

In the process of analyzing the best associated products placed together in a basket, the primary focus lies in identifying those items that are commonly purchased in pairs or combinations. To achieve this, the code snippet you provided, namely basket plus['1.5V IND AA ALK BULK'].sum(), serves the purpose of calculating the total quantity of '1.5V IND AA ALK BULK' items sold within the dataset, yielding a substantial figure of 1022 units. Similarly, the analysis extends to other products as well. For instance, it meticulously computes the total sales for '1.5V IND AAA ALK BULK,' revealing a count of 866 units sold. Moreover, it diligently tracks the performance of 'D ALKALINE BULK,' demonstrating a respectable sales volume of 622 units. Lastly, the code evaluates the total sales for '1.5V ALKALINE AA CONTRACTOR 24PK,' an item that garnered a commendable 715 units in sales. These numerical values signify the distinct popularity and individual sales volumes of each product within the given dataset. From this comprehensive examination, it becomes apparent that '1.5V IND AA ALK BULK' emerges as the most frequently sold item among the listed options. Following closely behind are '1.5V IND AAA ALK BULK' and '1.5V ALKALINE AA CONTRACTOR 24PK,' both of which exhibit strong sales performance leader in terms of frequency of purchase within the dataset. In summary, the detailed analysis presented here offers valuable insights into the relative popularity and sales figures of the mentioned products, highlighting '1.5V IND AA ALK BULK' as the standout.

```
:
basket_plus['1.5V IND AA ALK BULK'].sum()
:
1022.0
:
basket_plus['1.5V IND AAA ALK BULK'].sum()
:
806.0
:
basket_plus['D ALKALINE BULK'].sum()
:
622.0
:
basket_plus['1.5V ALKALINE AA CONTRACTOR 24PK'].sum()
:
715.0
:
basket_plus['6V GC2 ULTRA GOLF 12'].sum()
:
1896.0
```

Figure Demonstration

The last basket listed in the image refers to the sum of quantities for a product categorized as '6V GC2 ULTRA GOLF 12'. The .sum() function is used to add up all the values that fall under this specific category within the basket_plus dataset. The resulting sum for this category is 1896.0 units

The intention behind this analysis might be to identify which items are most purchased, which can inform various marketing strategies such as bundling (offering items together at a discounted rate), placement (arranging items near each other in a store to increase the likelihood of co-purchase), or promotions (targeting advertising to those items most likely to be bought). This demonstration not only highlights the individual performance of each item but also sets the stage for deeper analysis into potential associations between items and customer purchasing behaviour.

Strength

One of the standout strengths of association rule learning algorithms is their simplicity and ease of understanding, which democratizes data science by making these algorithms accessible to a broader spectrum of users, from seasoned data scientists to business analysts. The consistency of the results these algorithms produce is another merit, ensuring reliable output that businesses can trust for making informed decisions. Moreover, the intuitive nature of the rules generated facilitates effortless communication of insights to end-users, often crystallizing complex data patterns into clear, actionable strategies. The clustering in the context of Market Basket Analysis lies in its straightforward logic, consistent results, and the production of clear, easily communicable rules. These characteristics make it not only a powerful tool for data analysts but also an asset for business decision-makers seeking to apply data-driven insights to real-world scenarios.

Findings and Limitations

The comparative analysis between the Apriori and FP-Growth algorithms reveals some intriguing insights into the performance and efficiency of these popular association rule learning methods. The study highlighted that the Apriori algorithm, despite its widespread use and foundational status in the field of data mining, tends to require more time to complete its analysis. This is largely due to its methodical approach to combing through datasets, examining each combination of items to identify frequent itemsets, which can be time-consuming, especially as the size of the dataset expands. On the other hand, the FP-Growth algorithm stands out for its remarkable speed and efficiency in sorting and analyzing data. This can be attributed to its innovative use of the FP-tree structure, which condenses the dataset into a compact form, significantly reducing the number of scans required to identify frequent itemsets. This characteristic makes FP-Growth particularly well-suited for handling large datasets, as it can rapidly process vast amounts of data without the need for multiple database scans, thereby accelerating the analysis process.

Despite these differences in operational efficiency, both algorithms ultimately demonstrate a similar capability in deriving meaningful association rules from the datasets they analyze. This indicates that the choice between using Apriori or FP-Growth may often come down to specific considerations of

dataset size and analysis speed, rather than the quality of the resulting insights. However, the study also noted a limitation in the available data, particularly the lack of demographic and other relevant information that could enrich predictive modeling efforts. This gap underscores the importance of comprehensive data collection in maximizing the potential of data mining algorithms to generate actionable and insightful predictive models. Furthermore, the utility of Apriori and FP-Growth algorithms is not without its challenges, especially when applied to extremely large transactional datasets. One of the critical constraints encountered in such situations is the capacity of memory. As the volume of transactions increases, the memory required to efficiently process and analyze the data can exceed available resources, leading to a bottleneck that adversely affects the performance of these algorithms. This limitation is particularly pronounced in environments where memory is limited, and the scale of data analysis is extensive, raising concerns about the viability of these methods for certain large-scale applications.

Recommandation:

One of the most important the marketers could do, since the sales dropped after the Covid-19 they can promote the product to the target segment. For instance, they can hire SEO specialist and promote the products to the people who works from home and surf on webs. Moreover, creating communities with the young stars those who have passion in Remote Control games like RC-cars, planes, and other interesting things that will create a bond with the brand and ultimately will have emotional connection with the brand that ultimately will increase revenue.

Future research Directions

Plunging deeper into how artificial intelligence (AI) can enhance Market Basket Analysis (MBA) is a promising path. AI has the potential to unlock more detailed insights into customer preferences and to predict future buying patterns with greater accuracy, offering businesses a powerful tool to tailor their strategies. The possibility of applying Market Basket Analysis in real-time presents an exciting opportunity for businesses to react instantly to shifts in consumer interests, keeping their marketing strategies agile and aligned with current trends. Moreover, the role of MBA in improving the design and usability of e-commerce websites is another area developed for exploration. By understanding how customers shop, businesses can create more engaging and user-friendly online spaces that encourage more clicks and conversions. The emergence of new technologies like blockchain and the Internet of Things (IoT) also opens new avenues for making MBA more effective and integrated, potentially revolutionizing how data is collected and analysed in the e-commerce sector.

Furthermore, as sustainability becomes a priority for consumers and companies alike, investigating how MBA can support eco-friendly business practices in the e-commerce industry could lead to more sustainable shopping solutions. Additionally, the expanding landscape of multichannel retailing and the diverse shopping habits across global markets present a complex puzzle. Market Basket analysis could provide valuable insights into tailoring strategies to fit the varied shopping environments and

cultural preferences, ensuring that businesses stay relevant and responsive to their diverse customer base.

Conclusion

In short, this paper has articulated the robust capabilities of Market Basket Analysis (MBA) as an necessary tool in the domain of e-commerce and retail. By leveraging the Clustering and FP-Growth algorithms, along with Association Rules, the study has demonstrated a methodical approach to unravelling complex consumer purchasing patterns. The algorithms' facility for simplicity, paired with their consistency and intuitive output, democratizes the power of data analytics, making these insights accessible to a broad spectrum of users, from seasoned data scientists to business strategists. The core findings of this research underscore the significance of website quality in influencing online shopping behaviour, suggesting a pivotal shift towards user experience in e-commerce platforms. The paper also casts light on the efficiency of the FP-Growth algorithm over the Apriori algorithm.

By implementing these analytical techniques, the study has provided e-commerce companies with actionable strategies to optimize their marketing tactics and become more competitive in the market. The practical applications of the research are manifold, ranging from enhancing product placement to devising targeted promotions and crafting personalized customer engagement strategies.

8 Appendices

Data Loading

```
# Import Libraries for association rule mininig to find frequent patterns
import pandas as pd
import numpy as np


# Import Libraries for Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import re
import networkx as nx
```

```
#Taking the CSV file
df = pd.read_csv('Market Basket ANALYSIS').csv', encoding='latin')
```

```
# Check the first 10 records
df.head(10)
```

	Purchase Date	Product	Product Category	Transaction	Quantity	Product Revenue
0	2021-08-29 02:42:55.182481760	3.7V 3400mah LIION 12.6WH	Battery/Consumer Rechargeable	EC0043605902	47	1597.53
1	2021-08-25 02:48:10.510948912	3V PHOTO LITHIUM	Battery/Primary Other	EC0043507670	47	1246.44
2	2021-08-26 10:46:25.401459856	12V 11.2AH 225CCA AGM 12/0	Battery/Powersports	EC0043504182	41	4714.59
3	2020-05-03 21:17:04.817518248	12V 12AH 165CCA FLOODED 6/0	Battery/Powersports	EC0043503186	39	2456.61
4	2021-08-25 02:48:10.510948912	12V 12AH 210CCA AGM 12/0	Battery/Powersports	EC0043406547	34	3570.00
5	2020-05-07 21:11:49.489051096	12V 12AH 220CCA AGM	Battery/Powersports	EC0043406547	34	4759.66
6	2020-04-28 13:24:05.255474456	6V 4AH LEAD OE JR BLUE BATT FISHER PRICE	Battery/Sealed Lead Acid	EC0042903780	29	1507.71
7	2020-05-02 13:18:49.927007304	12V 31 MARINE DC 12	Battery/Marine/RV	EC0043607259	26	8579.48
8	2019-01-01 00:00:00.000000000	6V 2100MAH NIMH	Battery/Digital Camera/Camcorder	EC0043506900	26	779.74
9	2021-08-26 10:46:25.401459856	3.6V 1200MAH LITHI	Battery/Primary Other	EC0043208361	25	749.00

```
#checking data frame info

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92250 entries, 0 to 92249
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Purchase Date      92250 non-null  object
 1   Product            92250 non-null  object
 2   Product Category   92250 non-null  object
 3   Transaction        92250 non-null  object
 4   Quantity           92250 non-null  int64
 5   Product Revenue    92250 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 4.2+ MB
```

```
# Convert the 'Product Revenue' column to numeric, removing any non-numeric characters
df['Product Revenue'] = df['Product Revenue'].replace('[\$,]', '', regex=True).astype(float)
```

```
# Statistical analysis of data
df.describe()
```

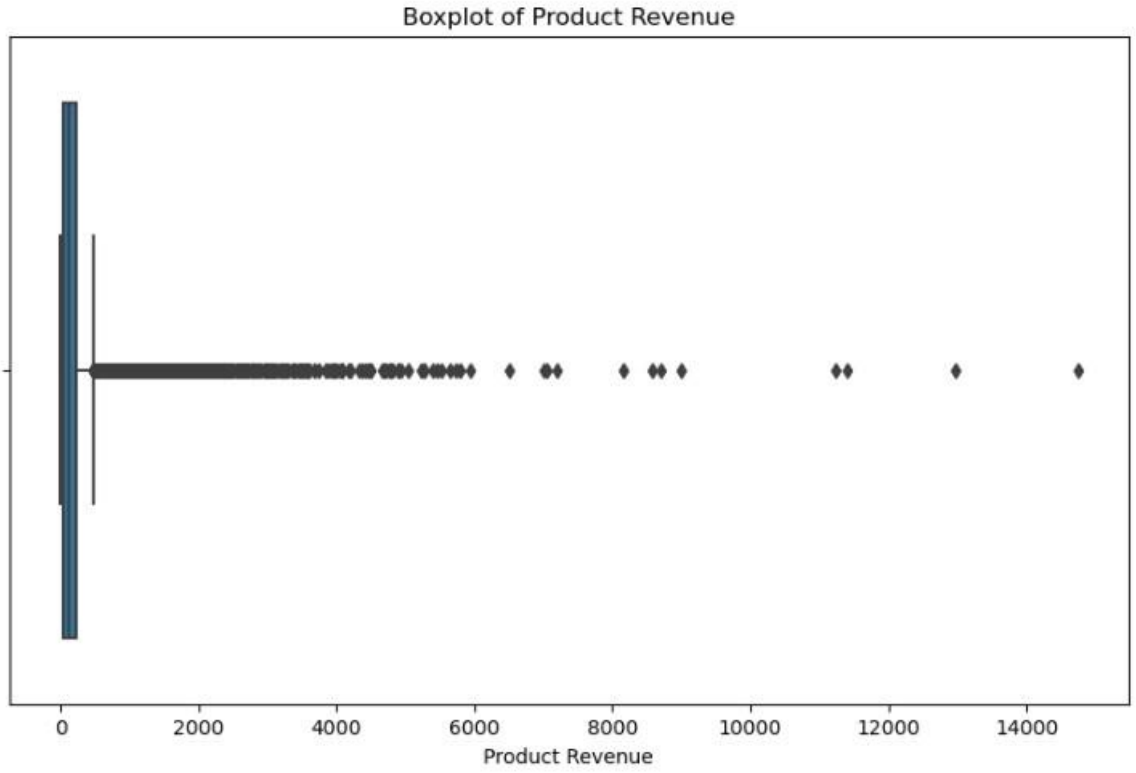
	Quantity	Product Revenue
count	92250.000000	92250.000000
mean	1.164564	190.383006
std	0.676474	297.407428
min	1.000000	0.000000
25%	1.000000	44.990000
50%	1.000000	116.990000
75%	1.000000	219.980000
max	47.000000	14739.200000

```
# find the no.  of rows and columns in dataframe respectively
df.shape
```

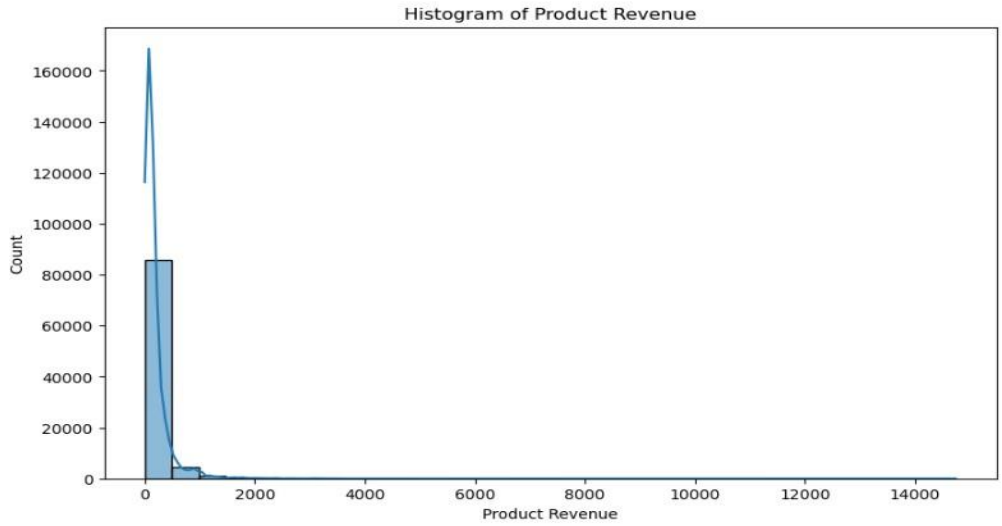
```
(92250, 6)
```

Visualization

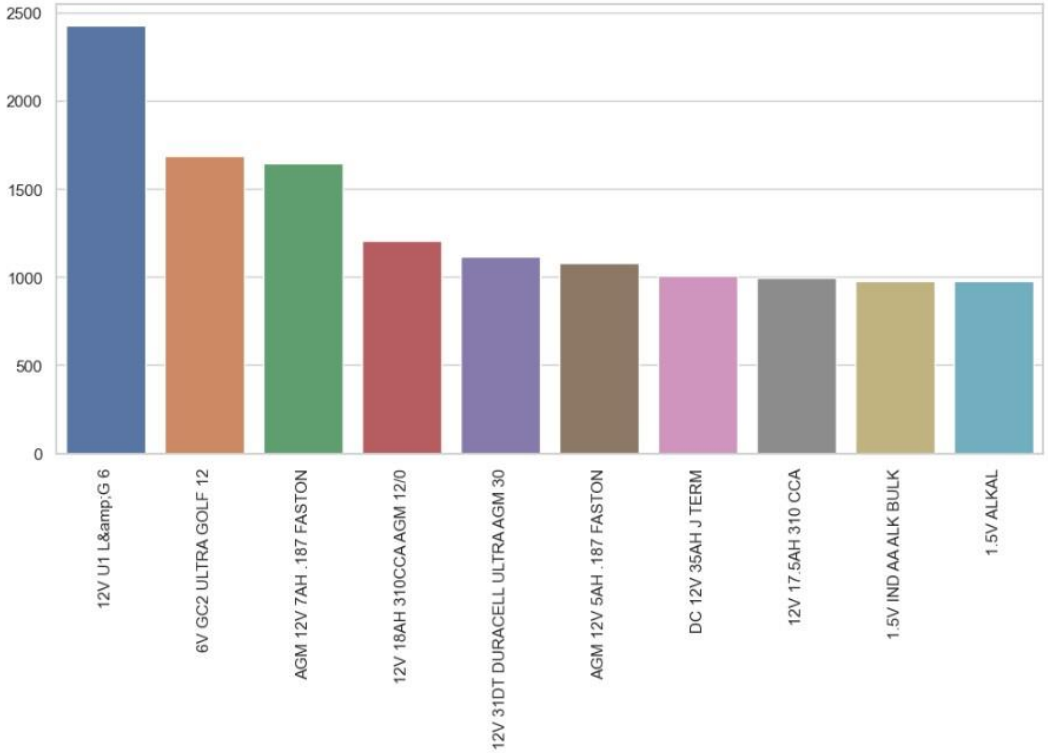
```
# Boxplot for 'Product Revenue'
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Product Revenue'])
plt.title('Boxplot of Product Revenue')
plt.show()
```



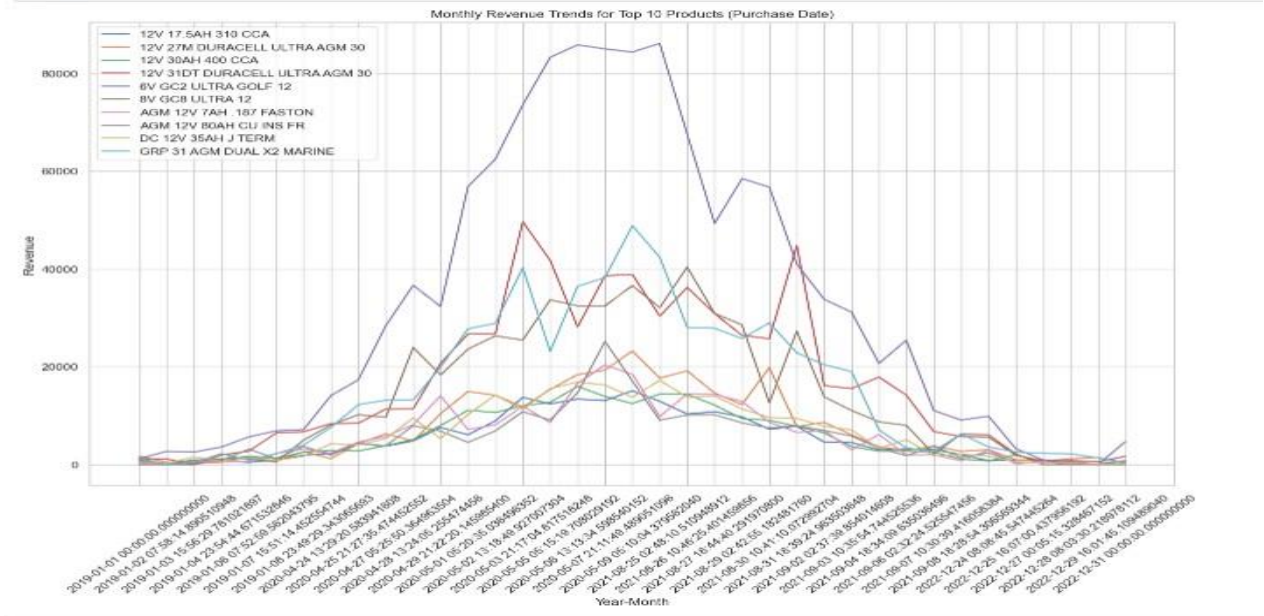

```
# Histogram for 'Product Revenue'
plt.figure(figsize=(10, 6))
sns.histplot(df['Product Revenue'], bins=30, kde=True)
plt.title('Histogram of Product Revenue')
plt.show()
```



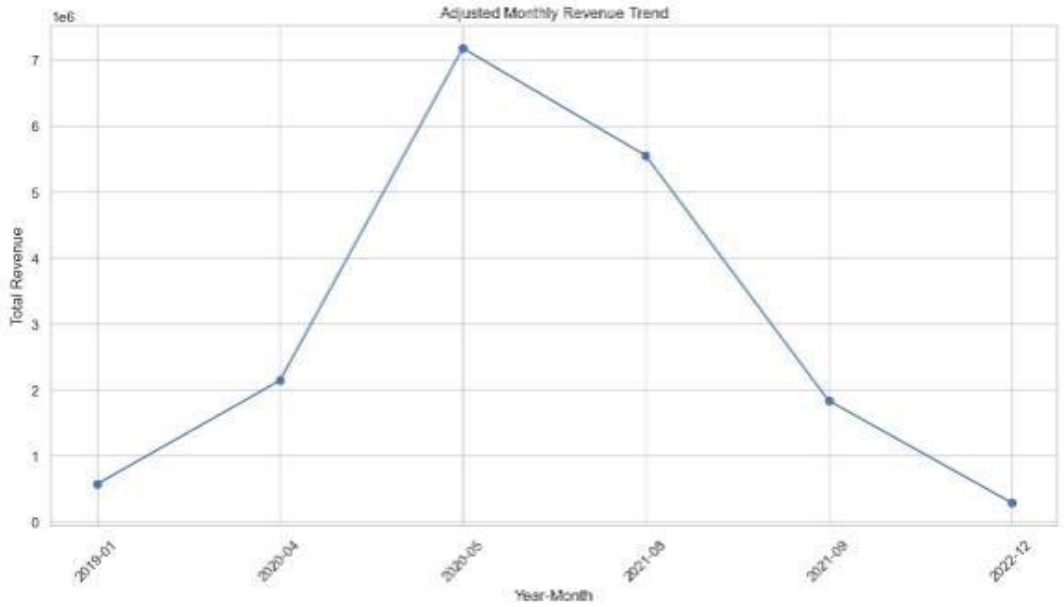
```
# Assuming 'x' is the series obtained from value_counts
x = df['Product'].value_counts().sort_values(ascending=False)[:10]

plt.figure(figsize=(12,5.5))
sns.barplot(x=x.index, y=x.values)
plt.xticks(rotation=90)
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'df' is your dataset and it's already loaded
# Aggregate product revenue to identify the top 10 products
product_revenue = df.groupby('Product')['Product Revenue'].sum().sort_values(ascending=False)

# Identify the top 10 products by total revenue
top_10_products = product_revenue.head(10).index.tolist()

# Now you can proceed with filtering the dataset for these top 10 products and plotting
top_10_data_adjusted = df[df['Product'].isin(top_10_products)]
monthly_revenue_top_10_adjusted = top_10_data_adjusted.groupby(['Purchase Date', 'Product'])['Product Revenue'].sum().unstack(fill_value=0)

# Plotting
plt.figure(figsize=(15, 10))
for product in monthly_revenue_top_10_adjusted.columns:
    plt.plot(monthly_revenue_top_10_adjusted.index.astype(str), monthly_revenue_top_10_adjusted[product], label=product)

plt.title('Monthly Revenue Trends for Top 10 Products (Purchase Date)')
plt.xlabel('Year-Month')
plt.ylabel('Revenue')
plt.xticks(rotation=45)
plt.legend(loc='upper left')
plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame and it's already loaded

# Convert 'Purchase Date' to datetime format
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'])

# Extract the month and year from the adjusted purchase dates
df['Adjusted Year-Month'] = df['Purchase Date'].dt.to_period('M')

# Aggregating total revenue by adjusted year-month
monthly_revenue_adjusted = df.groupby('Adjusted Year-Month')['Product Revenue'].sum().reset_index()

# Plotting the adjusted monthly revenue trend
plt.figure(figsize=(14, 7))
plt.plot(monthly_revenue_adjusted['Adjusted Year-Month'].astype(str), monthly_revenue_adjusted['Product Revenue'], marker='o', linestyle='-')
plt.title('Adjusted Monthly Revenue Trend')
plt.xlabel('Year-Month')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Data Cleaning

```
# Remove duplicates
df = df.drop_duplicates()
```

```
# Check for missing values
print(df.isnull().sum())
```

```
Purchase Date       0
Product             0
Product Category    0
Transaction         0
Quantity            0
Product Revenue     0
dtype: int64
```

```
df.drop(df[df['Transaction'] == '<transaction id>'].index, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 92249 entries, 0 to 92249
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Purchase Date     92249 non-null  object
 1   Product           92249 non-null  object
 2   Product Category  92249 non-null  object
 3   Transaction       92249 non-null  object
 4   Quantity          92249 non-null  int64
 5   Product Revenue   92249 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 4.9+ MB
```

```
# Check for duplicate rows
duplicate_rows = df.duplicated()
print(f"Number of duplicate rows = {duplicate_rows.sum()}")
```

```
Number of duplicate rows = 0
```

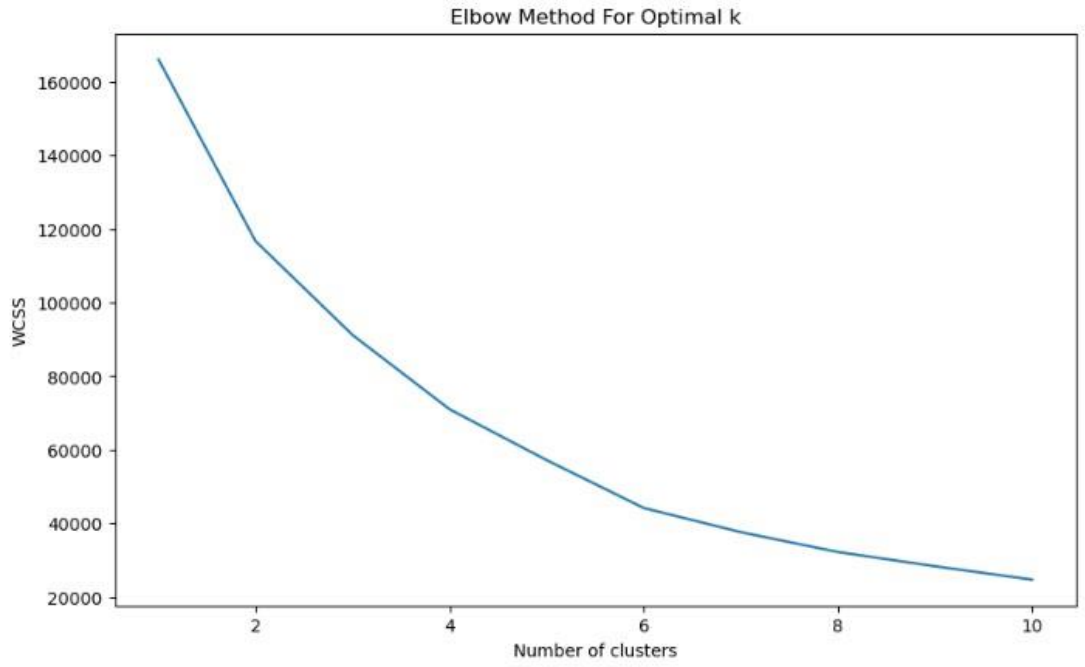
```
#taking only the posetive values of Quantity
df = df[df['Quantity']>=0]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 92249 entries, 0 to 92249
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Purchase Date     92249 non-null  object
 1   Product           92249 non-null  object
 2   Product Category  92249 non-null  object
 3   Transaction       92249 non-null  object
 4   Quantity          92249 non-null  int64
 5   Product Revenue   92249 non-null  float64
dtypes: float64(1), int64(1), object(4)
memory usage: 4.9+ MB
```

```
#picking up the top 20 products
z = df['Product'].value_counts().sort_values(ascending=False)[:20]
z
```

```
12V U1 L&amp;G 6                2425
6V GC2 ULTRA GOLF 12            1684
AGM 12V 7AH .187 FASTON         1642
12V 18AH 310CCA AGM 12/0        1204
12V 31DT DURACELL ULTRA AGM 30  1118
AGM 12V 5AH .187 FASTON         1077
DC 12V 35AH J TERM              1005
12V 17.5AH 310 CCA               996
1.5V IND AA ALK BULK             979
1.5V ALKAL                       977
12V 30AH 400 CCA                 940
12V U1R L&amp;G 6                928
1.55V SVROX                      925
12V 27 MARINE DP 12              922
12V 24 MARINE DP 12              900
GRP 31 AGM DUAL X2 MARINE        816
12V 24F ULTRA GOLD 36            811
1.5V IND AAA ALK BULK            789
AGM 12V 12AH .250 FASTON         781
12V 65 ULTRA 24                  751
Name: Product, dtype: int64
```

```
# Data Preprocessing
# Convert 'Product Revenue' to numeric by removing the dollar sign and commas
df['Product Revenue'] = df['Product Revenue'].replace('[\$,]', '', regex=True).astype(float)

# Aggregating data at the transaction level
# Summing up the quantity and revenue for each transaction
transaction_data = df.groupby('Transaction').agg({'Quantity': 'sum', 'Product Revenue': 'sum'}).reset_index()

# Feature Engineering
# Standardizing the data for clustering
scaler = StandardScaler()
scaled_features = scaler.fit_transform(transaction_data[['Quantity', 'Product Revenue']])

# Finding the optimal number of clusters using the Elbow Method
wcss = []  # Within-cluster sum of squares
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)

# Plotting the results to observe the 'Elbow'
plt.figure(figsize=(10,6))
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method For Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



```
# Assuming you've determined the optimal number of clusters is 'optimal_k'
optimal_k = 3  # replace with the actual number you've determined
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300, n_init=10, random_state=0)
kmeans.fit(scaled_features)

# Assign the cluster labels to each transaction
transaction_data['Cluster'] = kmeans.labels_

# Merge the cluster labels back to the original dataframe
df = df.merge(transaction_data[['Transaction', 'Cluster']], on='Transaction', how='left')

# Now you can examine which products are in each cluster
clustered_products = df.groupby('Cluster')['Product'].unique()

# To display the product names for each cluster
for cluster in range(optimal_k):
    print(f"Cluster {cluster}: {clustered_products[cluster]}")
```

```
Cluster 0: ['1.2V 1100MAH NICAD' '1.2V 12000MAH NIMH' '1.2V 1200MAH NICAD 4/5A' ...
 'YETI 200X PORTABLE POWER' 'ZBUG LANTERN + LIGHT'
 'ZUS UNIVERSAL CAR AUDIO ADAPTER']
Cluster 1: ['3.7V 3400mah LIION 12.6WH' '3V PHOTO LITHIUM'
 '12V 11.2AH 225CCA AGM 12/0' '12V 12AH 165CCA FLOODED 6/0'
 '12V 12AH 210CCA AGM 12/0' '12V 12AH 220CCA AGM'
 '6V 4AH LEAD OE JR BLUE BATT FISHER PRICE' '12V 31 MARINE DC 12'
 '6V 2100MAH NIMH' '3.6V 1200MAH LITHI' '12.8V 3750MAH LIFEPO4 48W'
 '12V 26AH 400CCA AGM' '12V 27 ULTRA GOLD 36' '12V 48 ULTRA PLATINUM 48'
 'REMOTE FORD' '12V 24 MARINE DP 12' '2PK 50Q/CL/MC'
 'CHRYSLER DODGE JEEP REMOTE' 'CORD BYTES UNICORN/PANDA'
 '22.2V 2200MAH LIION 48.8W' 'GRP 34 AGM DUAL X2 MARINE' '12V 35 ULTRA 24'
 '3.2V 600MAH LIFEPO4 1.9WH' 'GRP 31 AGM DUAL X2 MARINE'
 'OEM replacement battery for baby monitors' '12V 27 MARINE DP 12'
 '12V 8AH 185CCA AGM 12/0' '12V 18AH 310CCA AGM 12/0'
 '12V 48 AGM DUAL X2 AUTO 60' '12V 94R ULTRA GOLD 36'
 '4 BUTTON REMOTE HEAD KEY REPLACEMENT FOR HONDA' '12V 51R ULTRA GOLD 36'
 '12V MIATA ULTRA 24' '1.2V 700MAH NICAD AA' '1.55V SVROX'
 '12V 10AH 175CCA AGM 12/0' '12V 11AH 205CCA AGM 12/0' '12V 121R ULTRA 24'
```

```
# Merge the cluster labels with the original dataframe
df_merged = df.merge(transaction_data[['Transaction', 'Cluster']], on='Transaction', how='left')

# Group by 'Cluster' and 'Product', count the occurrences, and reset the index
clustered_product_counts = df.groupby(['Cluster', 'Product']).size().reset_index(name='counts')

# Sort the counts within each cluster and get the top 5
top_clustered_products = clustered_product_counts.groupby('Cluster').apply(lambda x: x.nlargest(5, 'counts')).reset_index(drop=T

# Display the top 5 products for each cluster
top_clustered_products
```

```
C:\Users\220372600\AppData\Local\Temp\ipykernel_10824\2624076994.py:2: FutureWarning: Passing 'suffixes' which cause duplicate
columns {'Cluster_x'} in the result is deprecated and will raise a MergeError in a future version.
  df_merged = df.merge(transaction_data[['Transaction', 'Cluster']], on='Transaction', how='left')
```

	Cluster	Product	counts
0	0	12V U1 L&G 6	2308
1	0	AGM 12V 7AH .187 FASTON	1443
2	0	12V 18AH 310CCA AGM 12/0	1136
3	0	AGM 12V 5AH .187 FASTON	998
4	0	DC 12V 35AH J TERM	901
5	1	6V GC2 ULTRA GOLF 12	29
6	1	AGM 12V 7AH .187 FASTON	19
7	1	AGM 12V 12AH .250 FASTON	17
8	1	8V GC8 ULTRA 12	16
9	1	12V 31DT DURACELL ULTRA AGM 30	15
10	2	6V GC2 ULTRA GOLF 12	759
11	2	1.5V IND AA ALK BULK	457
12	2	1.5V IND AAA ALK BULK	438
13	2	8V GC8 ULTRA 12	399
14	2	1.5V ALKAL	360

Creating Basket ¶

```
# Define a function 'encode_units' that takes a value 'x'
def encode_units(x):
    # If 'x' is less than or equal to 0, return 0
    if x <= 0:
        return 0
    # If 'x' is greater than or equal to 1, return 1
    if x >= 1:
        return 1

# Apply the 'encode_units' function to each element in the 'basket_plus' DataFrame
basket_encode_plus = basket_plus.applymap(encode_units)

# 'basket_encode_plus' now contains the result of applying 'encode_units' to each element in 'basket_plus'
# This code converts the original values in 'basket_plus' to 1 if they were greater than or equal to 1,
# and to 0 if they were less than or equal to 0.
```

Product	CENTIUM IS UNV	... set)	ONBOARD BATTERY CHARGER	ONBOARD BATTERY CHARGER	1100MAH NICAD	12000MAH NIMH	1200MAH NICAD	NICAD 4/5A	1200MAH NIMH	NICAD 4/5A	...	W/METAL JACKET	W/METAL JACKET	POWE STATIO
Transaction														
EC0032704676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
EC0042407100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
EC0042508772	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
EC0042607412	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
EC0042609230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
EC0044007294	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
EC0044007295	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
EC20212935	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.
[illegible]	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.

```
#dropping off not set column

basket_plus.drop('(not set)', axis=1, inplace=True)
basket_plus
```

Product	(4)F32T8 CENTIUM IS UNV	1 BANK 10A ONBOARD BATTERY CHARGER	1 BANK 5A ONBOARD BATTERY CHARGER	1.2V 1100MAH NICAD	1.2V 12000MAH NIMH	1.2V 1200MAH NICAD	1.2V 1200MAH NICAD 4/5A	1.2V 1200MAH NIMH	1.2V 1400MAH NICAD 4/5A	1.2V 1500MAH NIMH	...	Y50-N18L-A W/METAL JACKET	YB16L-B W/METAL JACKET	YETI POR P ST
Transaction														
EC0032704676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
EC0042407100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
EC0042508772	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
EC0042607412	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
EC0042609230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
EC0044007294	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
[illegible]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	

```
# Define a function 'encode_units' that takes a value 'x'
def encode_units(x):
    # If 'x' is less than or equal to 0, return 0
    if x <= 0:
        return 0
    # If 'x' is greater than or equal to 1, return 1
    if x >= 1:
        return 1

# Apply the 'encode_units' function to each element in the 'basket_plus' DataFrame
basket_encode_plus = basket_plus.applymap(encode_units)

# The 'basket_encode_plus' DataFrame now contains binary values (0 or 1).
# - If an element in 'basket_plus' was less than or equal to 0, it is converted to 0.
# - If an element in 'basket_plus' was greater than or equal to 1, it is converted to 1.
# This code effectively transforms the original numeric values in 'basket_plus' into binary form.
```

Product	(4)F32T8 CENTIUM IS UNV	1 BANK 10A ONBOARD BATTERY CHARGER	1 BANK 5A ONBOARD BATTERY CHARGER	1.2V 1100MAH NICAD	1.2V 12000MAH NIMH	1.2V 1200MAH NICAD	1.2V 1200MAH NICAD 4/5A	1.2V 1200MAH NIMH	1.2V 1400MAH NICAD 4/5A	1.2V 1500MAH NIMH	...	Y50-N18L-A W/METAL JACKET	YB16L-B W/METAL JACKET	YETI 150 PORTAB POW STATI
Transaction														
1234	0	0	0	0	0	0	0	0	0	0	...	0	0	
123456	0	0	0	0	0	0	0	0	0	0	...	0	0	
12345678	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0032704676	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0042407100	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
EC0044007291	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007292	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007293	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007294	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007295	0	0	0	0	0	0	0	0	0	0	...	0	0	

83013 rows × 2413 columns

```
#add in the methodology
#In market basket analysis, the objective is to discover the relationships between two or more items that have been purchased to
#(Bought More Than 1 Items Only)
basket_filter_plus = basket_encode_plus[(basket_encode_plus > 0).sum(axis=1) >= 2]
basket_filter_plus
```

```
#add in the methodology
#In market basket analysis, the objective is to discover the relationships between two or more items that have been purchased to
#(Bought More Than 1 Items Only)
basket_filter_plus = basket_encode_plus[(basket_encode_plus > 0).sum(axis=1) >= 2]
basket_filter_plus
```

Product	(4)F32T8 CENTIUM IS UNV	1 BANK 10A ONBOARD BATTERY CHARGER	1 BANK 5A ONBOARD BATTERY CHARGER	1.2V 1100MAH NICAD	1.2V 12000MAH NIMH	1.2V 1200MAH NICAD	1.2V 1200MAH NICAD 4/5A	1.2V 1200MAH NIMH	1.2V 1400MAH NICAD 4/5A	1.2V 1500MAH NIMH	...	Y50-N18L-A W/METAL JACKET	YB16L-B W/METAL JACKET	YETI 150 PORTAB POW STATI
Transaction														
EC0032704676	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0042801384	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0042802882	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0042803407	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0042803469	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
EC0044007261	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007268	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007269	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007281	0	0	0	0	0	0	0	0	0	0	...	0	0	
EC0044007294	0	0	0	0	0	0	0	0	0	0	...	0	0	

6321 rows × 2413 columns

Applying the FP Growth and Apriori

FP GROWTH

```
!pip install mlxtend

from mlxtend.frequent_patterns import fpgrowth
# Apply FP-Growth algorithm
frequent_itemsets = fpgrowth(basket_filter_plus, min_support=0.03, use_colnames=True)
# Add a new column 'length' that stores the length of each itemset
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
# Print the frequent itemsets along with their lengths
print(frequent_itemsets)
```

```
Requirement already satisfied: mlxtend in c:\programdata\anaconda3\lib\site-packages (0.23.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (3.7.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.2.1)
Requirement already satisfied: numpy>=1.16.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.23.5)
Requirement already satisfied: joblib>=0.13.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.1.1)
Requirement already satisfied: scipy>=1.2.1 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.10.0)
Requirement already satisfied: pandas>=0.24.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.5.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(22.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxt
end) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(9.4.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (1.4.4)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.
11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(1.0.5)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2022.
7)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=1.0.2->ml
xtend) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=
3.0.0->mlxtend) (1.16.0)
```

```
C:\ProgramData\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with non-b
ool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame
with bool type
  warnings.warn(
```

	support	itemsets	length
0	0.098719	(1.5V ALKAL)	1
1	0.079418	(1.5V ALKALINE AA CONTRACTOR 24PK)	1
2	0.111691	(1.5V IND AAA ALK BULK)	1
3	0.056320	(D ALKALINE BULK)	1
4	0.042240	(9V IND ALK BULK)	1
5	0.115013	(1.5V IND AA ALK BULK)	1
6	0.030059	(1.5V ALKALINE D CONTRACTOR 12PK)	1
7	0.036228	(C ALKALINE BULK)	1
8	0.037810	(24PK 1.5V AA ALKALINE)	1
9	0.051732	(1.5V ALKAL, 1.5V ALKALINE AA CONTRACTOR 24PK)	2
10	0.086062	(1.5V IND AAA ALK BULK, 1.5V IND AA ALK BULK)	2
11	0.030059	(1.5V IND AAA ALK BULK, D ALKALINE BULK)	2
12	0.031641	(D ALKALINE BULK, 1.5V IND AA ALK BULK)	2

Apriori

```
from mlxtend.frequent_patterns import apriori
frequent_itemsets_plus = apriori(basket_filter_plus, min_support=0.03,
 use_colnames=True).sort_values('support', ascending=False).reset_index(drop=True)
frequent_itemsets_plus['length'] = frequent_itemsets_plus['itemsets'].apply(lambda x: len(x))
frequent_itemsets_plus
```

```
C:\ProgramData\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with non-b
ool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame
with bool type
  warnings.warn(
```

	support	itemsets	length
0	0.115013	(1.5V IND AA ALK BULK)	1
1	0.111691	(1.5V IND AAA ALK BULK)	1
2	0.098719	(1.5V ALKAL)	1
3	0.086062	(1.5V IND AAA ALK BULK, 1.5V IND AA ALK BULK)	2
4	0.079418	(1.5V ALKALINE AA CONTRACTOR 24PK)	1
5	0.056320	(D ALKALINE BULK)	1
6	0.051732	(1.5V ALKAL, 1.5V ALKALINE AA CONTRACTOR 24PK)	2
7	0.042240	(9V IND ALK BULK)	1
8	0.037810	(24PK 1.5V AA ALKALINE)	1
9	0.036228	(C ALKALINE BULK)	1
10	0.031641	(D ALKALINE BULK, 1.5V IND AA ALK BULK)	2
11	0.030059	(1.5V ALKALINE D CONTRACTOR 12PK)	1
12	0.030059	(1.5V IND AAA ALK BULK, D ALKALINE BULK)	2

Frequently Bought Items 1 and 2

```
#items bought atleast one
frequent_itemsets[ (frequent_itemsets['length'] == 1)]
```

	support	itemsets	length
0	0.098719	(1.5V ALKAL)	1
1	0.079418	(1.5V ALKALINE AA CONTRACTOR 24PK)	1
2	0.111691	(1.5V IND AAA ALK BULK)	1
3	0.056320	(D ALKALINE BULK)	1
4	0.042240	(9V IND ALK BULK)	1
5	0.115013	(1.5V IND AA ALK BULK)	1
6	0.030059	(1.5V ALKALINE D CONTRACTOR 12PK)	1
7	0.036228	(C ALKALINE BULK)	1
8	0.037810	(24PK 1.5V AA ALKALINE)	1

Finding The Association Between Frequently Bought Items ¶

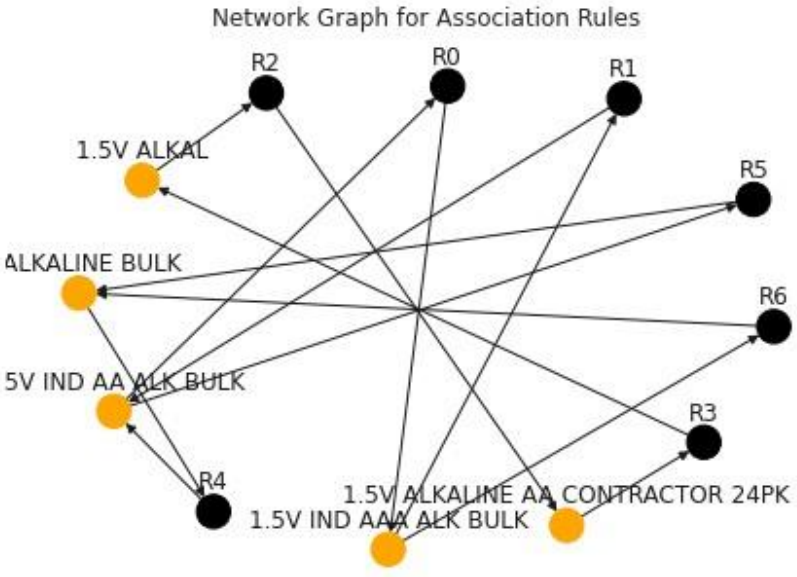
```
#finding relation of the associated products
from mlxtend.frequent_patterns import association_rules
rules = association_rules(frequent_itemsets, metric='lift',
 min_threshold=1).sort_values('lift', ascending=False).reset_index(drop=True)
DF=rules
DF
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(1.5V IND AA ALK BULK)	(1.5V IND AAA ALK BULK)	0.115013	0.111691	0.086062	0.748281	6.699549	0.073216	3.528965	0.961298
1	(1.5V IND AAA ALK BULK)	(1.5V IND AA ALK BULK)	0.111691	0.115013	0.086062	0.770538	6.699549	0.073216	3.856793	0.957703
2	(1.5V ALKAL)	(1.5V ALKALINE AA CONTRACTOR 24PK)	0.098719	0.079418	0.051732	0.524038	6.598500	0.043892	1.934152	0.941382
3	(1.5V ALKALINE AA CONTRACTOR 24PK)	(1.5V ALKAL)	0.079418	0.098719	0.051732	0.651394	6.598500	0.043892	2.585390	0.921645
4	(D ALKALINE BULK)	(1.5V IND AA ALK BULK)	0.056320	0.115013	0.031641	0.561798	4.884627	0.025163	2.019585	0.842739
5	(1.5V IND AA ALK BULK)	(D ALKALINE BULK)	0.115013	0.056320	0.031641	0.275103	4.884627	0.025163	1.301813	0.898631
6	(1.5V IND AAA ALK BULK)	(D ALKALINE BULK)	0.111691	0.056320	0.030059	0.269122	4.778424	0.023768	1.291159	0.890148
7	(D ALKALINE BULK)	(1.5V IND AAA ALK BULK)	0.056320	0.111691	0.030059	0.533708	4.778424	0.023768	1.905048	0.837918

3D Scatter Plot of rules

```
sns.set(style = "whitegrid")
fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(projection = '3d')
x = rules['support']
y = rules['confidence']
z = rules['lift']
ax.set_xlabel("Support")
ax.set_ylabel("Confidence")
ax.set_zlabel("Lift")
ax.scatter(x, y, z)
ax.set_title("3D Distribution of Association Rules")
plt.show()
```

```
def draw_network(rules, rules_to_show):

    # Directional Graph from Networkx
    network = nx.DiGraph()

    # Loop through number of rules to show
    for i in range(rules_to_show):
        # Add a Rule Node
        network.add_nodes_from(["R" + str(i)])
        for antecedents in rules.iloc[i]['antecedents']:
            # Add antecedent node and link to rule
            network.add_nodes_from([antecedents])
            network.add_edge(antecedents, "R" + str(i), weight = 2)

        for consequents in rules.iloc[i]['consequents']:
            # Add consequent node and link to rule
            network.add_nodes_from([consequents])
            network.add_edge("R" + str(i), consequents, weight = 2)

    color_map = []

    # For every node, if it's a rule, color as Black, otherwise Orange

    for node in network:
        if re.compile("^[R]\d+$").fullmatch(node)!=None:
            color_map.append('black')
        else:
            color_map.append('orange')

    # Position nodes using spring layout
    pos = nx.spring_layout(network, k=16, scale=1)

    # Draw the network graph
    nx.draw(network, pos, node_color = color_map, font_size = 8)

    # Shift the test position upwards
    for p in pos:
        pos[p][1] += 0.12

    nx.draw_networkx_labels(network, pos)
    plt.title("Network Graph for Association Rules")
    plt.show()

draw_network(rules, 7)
```



Association Rules with Most Strength

```
df_ms=rules[(rules['lift'] >= 6) &]
 (rules['confidence'] >= 0.5)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(1.5V IND AA ALK BULK)	(1.5V IND AAA ALK BULK)	0.115013	0.111691	0.086062	0.748281	6.699549	0.073216	3.528965	0.961298
1	(1.5V IND AAA ALK BULK)	(1.5V IND AA ALK BULK)	0.111691	0.115013	0.086062	0.770538	6.699549	0.073216	3.856793	0.957703
2	(1.5V ALKAL)	(1.5V ALKALINE AA CONTRACTOR 24PK)	0.098719	0.079418	0.051732	0.524038	6.598500	0.043892	1.934152	0.941382
3	(1.5V ALKALINE AA CONTRACTOR 24PK)	(1.5V ALKAL)	0.079418	0.098719	0.051732	0.651394	6.598500	0.043892	2.585390	0.921645

```
df_ms.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4 entries, 0 to 3
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   antecedents         4 non-null      object 
 1   consequents         4 non-null      object 
 2   antecedent support  4 non-null      float64
 3   consequent support  4 non-null      float64
 4   support             4 non-null      float64
 5   confidence          4 non-null      float64
 6   lift                4 non-null      float64
 7   leverage            4 non-null      float64
 8   conviction          4 non-null      float64
 9   zhangs_metric       4 non-null      float64
dtypes: float64(8), object(2)
memory usage: 352.0+ bytes
```

Checking the best two items together

```
basket_plus['1.5V IND AA ALK BULK'].sum()
```

```
1022.0
```

```
basket_plus['1.5V IND AAA ALK BULK'].sum()
```

```
806.0
```

```
basket_plus['D ALKALINE BULK'].sum()
```

```
622.0
```

```
basket_plus['1.5V ALKALINE AA CONTRACTOR 24PK'].sum()
```

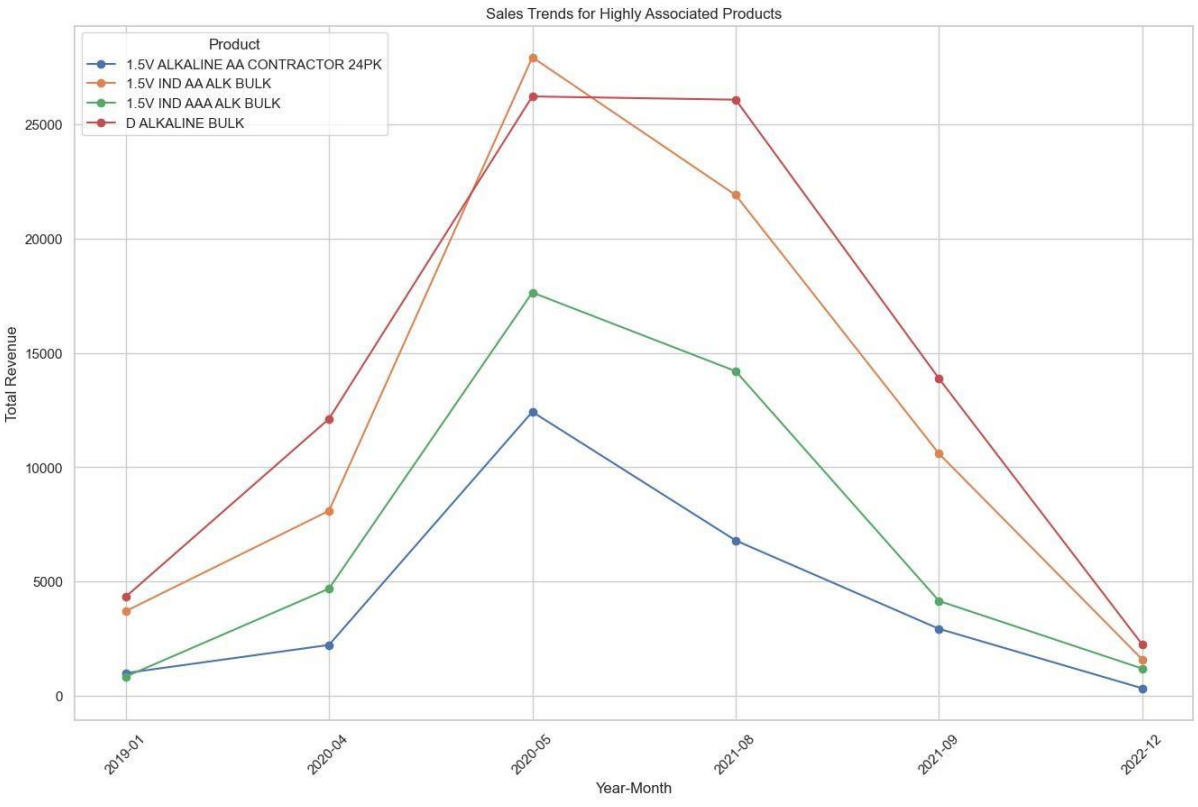
```
715.0
```

```
basket_plus['6V GC2 ULTRA GOLF 12'].sum()
```

```
1896.0
```

```python
# Define the list of highly associated products
associated_products = [
    "1.5V IND AAA ALK BULK",
    "1.5V IND AA ALK BULK",
    "D ALKALINE BULK",
    "1.5V ALKALINE AA CONTRACTOR 24PK"
]

# Filter the dataset for the selected products
filtered_data = df[df['Product'].isin(associated_products)]

# Ensure 'Purchase Date' is in datetime format for aggregation
filtered_data['Purchase Date'] = pd.to_datetime(filtered_data['Purchase Date'])

# Extract Year-Month for aggregation
filtered_data['Year-Month'] = filtered_data['Purchase Date'].dt.to_period('M')

# Aggregate sales data (revenue) by Year-Month for each product
aggregated_sales = filtered_data.groupby(['Year-Month', 'Product'])['Product Revenue'].sum().unstack(fill_value=0)

# Plotting the trends for the selected products
plt.figure(figsize=(16, 10))
for product in aggregated_sales.columns:
    plt.plot(aggregated_sales.index.astype(str), aggregated_sales[product], label=product, marker='o')

plt.title('Sales Trends for Highly Associated Products')
plt.xlabel('Year-Month')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.legend(title='Product', loc='upper left')
plt.grid(True)
plt.show()
```

```
C:\Users\220372600\AppData\Local\Temp\ipykernel_9272\4227201541.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_data['Purchase Date'] = pd.to_datetime(filtered_data['Purchase Date'])
C:\Users\220372600\AppData\Local\Temp\ipykernel_9272\4227201541.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_data['Year-Month'] = filtered_data['Purchase Date'].dt.to_period('M')
```



Bibliography

Aldino, A. A. et al., 2021. Comparison Of Market Basket Analysis To Determine Consumer Purchasing Patterns Using Fp-Growth And Apriori Algorithm. *International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE),* pp. 29-34.

Anon., 2019. Online Consumer Typologies and Their Shopping Behaviors in B2C E-Commerce Platforms. *Sage Journals,* 9(2), pp. 1-19.

Anon., n.d. *Gartner Glossary.* [Online]
Available at: https://www.gartner.com/en/information-technology/glossary/digitalization [Accessed 15 January 2024].

Arcos, D., R., J., A., A. & Hernandez, 2019. Analyzing Online Transaction Data using Association Rule. *Association for Computing Machinery,* pp. 43-49.

Chen, et al., 2005. Market basket analysis in a multiple store environment. *Decision support systems,* Volume 40(2), pp. 339-354.

Davis, F. D., 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly,* Volume 13(3), pp. 319-340.

Hani A. Jawa, K. C., 2015. Factors Influencing Consumer Behavior Towards Online. *International Journal of Multicultural and Multireligious Understanding,* 2(4), pp. 1-10.

Mansour, A. A. E., 2020. The Case of the 5 Largest E-Commerce Companies in the World. *The Effect of COVID-19 Spread on the E-Commerce Market,* p. 14.

Marcos, M., Bel´en, E. & Mar´ıa, E. G.-D. D. P. P.-R., 2021. Case Study: Appliances Sales Company. *Market basket analysis with association rules in the retail sector using Orange,* Volume 24.

Moodley, R., Francisco, C., Fabio, C. & Jenny, C., 2019. Application of uninorms to market basket analysis. *International Journal of Intelligent Systems,* Volume 34(1), pp. 39-49.

Mustapha, I., Mohammed Mansur, I., Zayyan Mahmoud, S. & Muesser, N., 2015. International Journal of Communications, Network and System Sciences. *Data Mining in Electronic Commerce: Benefits and Challenges,* Volume 08(12), p. 9.

Nakano, S., 2023. Customer demand concentration in online grocery retailing: Differences between online and physical store shopping baskets. *Electronic Commerce Research and Applications,* Volume 62.

Nanang, R., Muhammad Fariz, M. & Richard, S., 2019. Analysis of Customers Purchase Patterns of ECommerce Transactions Using Apriori Algorithm. *Scientific Research Journal (SCIRJ),* VII(VII), pp. 4558.

Petra, J. & Petra, K., 2020. *Advances in Economics, Business and Management Research,* Volume 131, pp. 630-635.

Rangga, Y., Ferdiansyah & Linda, A., 2022. Implementasi Algoritma Frequent Growth (FP-Growth) Menentukan Asosiasi. *Jurnal Sistem Komputer dan Informatika (JSON),* Volume 4, p. 165−171.

Raorane, A., Kulkarni, R. & & Jitkar, B., 2012. Association Rule – Extracting Knowledge Using Market Basket Analysis. *Research Journal of Recent Sciences,* Volume 1(2), pp. 19-27.

Riccardo, M., Gino, M., S. & Angela, T., 2015. A review of the environmental implications of B2C ecommerce: a logistics perspective. *International Journal of Physical Distribution & Logistics Management,* 45(6).

Ridwan, M., Suyono, H. & Sarosa, M., 2013. Penerapan Data Mining Untuk Evaluasi Kinerja Akademik Mahasiswa Menggunakan Algoritma Naive Bayes Classifier. *Jurnal EECCIS,* Volume 7(1), pp. 59-64.

Saikat, G. et al., 2020. Role of Artificial Intelligence (AI) in Understanding the Behavior Pattern: A Study on E-Commerce. *ICDSMLA 2019,* Volume 601, p. 1600–1606.

Santoso & M., H., 2021. Application of Association Rule Method Using Apriori Algorithm to Find. *Brilliance: Research of Artificial Intelligence,* Volume 1(2), pp. 1-13.

Setiabudi, D. H. & Budhi, G. S., n.d. Case Study in Minimarket X. *"Data Mining Market Basket Analysis.*

Soma, B., Thakur, S. & Mandal., K., 2021. Product recommendation for e-commerce business by applying principal component analysis (PCA) and K-means clustering: benefit for the society. *Innovations in Systems and Software Engineering,* Volume 17, pp. 45-52.

Sukenda, W. & Ari Purno, S., 2020. International Journal of Psychosocial Rehabilitation. *MEDICINE PRODUCT RECOMMENDATION SYSTEM USING APRIORI ALGORITHM AND FP-GROWTH ALGORITHM,* 24(02).

Ünvan & Yüksel, A., 2021. Market basket analysis with association rules. *Communications in Statistics-Theory and Methods,* Volume 50(7), pp. 1615-1628.

Xudong, S. et al., 2024. A scalable and flexible basket analysis system for big transaction. *Information Processing and Management,* Volume 225, pp. 4157- 4166.

Young-Chan, L., Liu, R.-Q. & Hong, -L. M., 2018. Customer Classification and Market Basket Analysis Using.

G., A. & A, E.-M. A., 2016. Understanding consumer intention to participate in online travel community and effects on consumer intention to purchase travel online and WOM: An integration of innovation diffusion theory and TAM with trust. *Computers in Human Behavior,* Volume 60, pp. 97111.