ISOM-676-4103: Machine Learning & AI at Scale

# ACSE Supermarket Recommender System: Recommender System Report

Team 3: Richard Jensen, Dylan Liu, Yifei Wang, Laurie Ye, Qing Ying

Apr. 15, 2024

## Table of Contents

**Executive Summary** 1

**I . Introduction** 2
1.1 Project Introduction 2
1.2 Brand Introduction 2

**II. Methodology** 3
2.1 Data Understanding 3
2.2 Data Preparation 3
2.3 Clustering 3
2.4 Association Rules 5
2.5 Logistic Regression 6

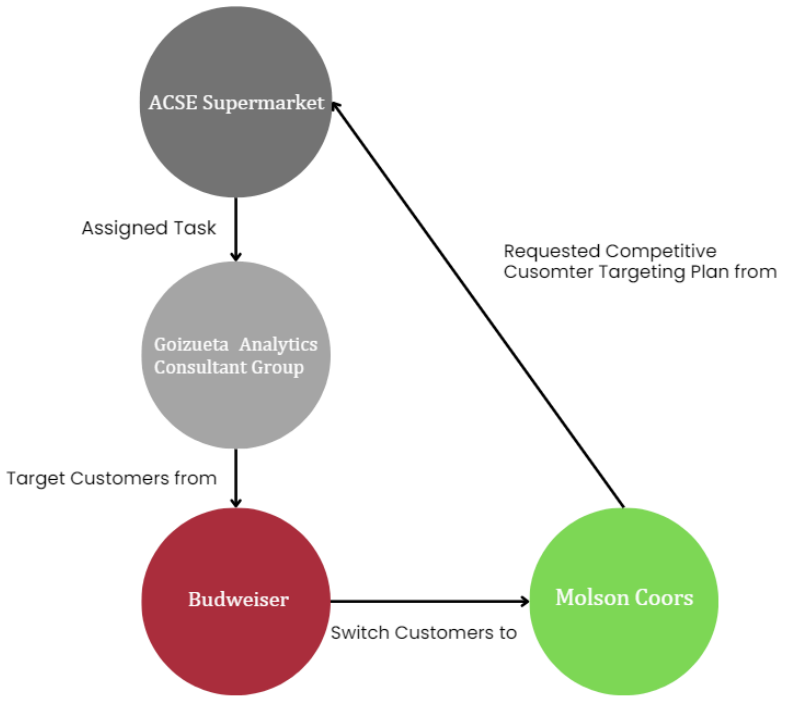
**III. Analysis** 8
3.1 Target Customer List 8
3.2 Predictions 9

**IV. Challenges and Conclusion** 11
4.1 Challenges 11
4.2 Conclusion 12

**Codebook Content** 13

## Executive Summary

As *Goizueta Analytics Consulting Group*, we have been commissioned by *ACSE* to execute a competitive customer targeting initiative for our client, *Molson Coors*. The project focuses on deploying targeted product recommendations aimed at *Budweiser*'s customer base, with the objective of shifting these customers to *Molson Coors*, thereby enhancing its market share in the long term.



To achieve this, we first developed a scalable recommender system utilizing transaction data from *ACSE*. We then specialized the system for the Domestic Beer category, employing customer clustering to identify distinct purchasing patterns among beer consumers and classify each customer according to their purchase history. This was followed by training logistic regression models to predict the likelihood of each customer in these clusters purchasing *Molson Coors* products. Each target customer is then assigned to a cluster, and the corresponding logistic regressions are used to determine their probability of purchasing each product, with the highest probability product being recommended.

While the system requires further refinement to include multi-product recommendations, adapt to changing customer preferences, and manage inventory efficiently, we believe that it closely aligns with *Molson Coors*' strategic goals. The system has successfully identified 42,000 potential customers, providing *Molson Coors* with detailed recommendations tailored to convert Budweiser's customers to their brand.

# I. Introduction

## 1.1 Project Introduction

**Background:** ACSE Supermarket, with over 40 locations in North America, offers more than 100,000 products across over 100 categories. The company plans to implement a recommender system to enhance inventory and marketing strategies. Moving away from general in-store promotions, ACSE is transitioning to personalized promotions through collaborations with select suppliers. These promotions are customized based on customer purchase history to increase marketing efficacy. The new recommender system will support this shift, enabling more targeted and effective promotional efforts.

**Objective:** Our team developed and implemented a recommender system for ACSE, specifically designed to analyze the potential of increasing Molson Coors product sales by targeting Budweiser customers.

## 1.2 Brand Introduction

Budweiser and Molson Coors are two major players in the beer industry.

**Molson Coors:** Molson Coors Beverage Company is one of the largest brewing companies in the world. It is known for various brands, including Coors, Molson Canadian, Blue Moon, and Miller Lite. Molson Coors has a diverse portfolio catering to different tastes and preferences within the beer market. Their products range from light lagers to craft beers and ales, allowing them to appeal to a broad consumer base.

**Budweiser:** Budweiser is a flagship brand of Anheuser-Busch InBev, another global brewing giant. It is one of the most recognizable beer brands globally, known for its iconic red label and Clydesdale horses. Budweiser primarily focuses on producing pale lagers, offering a consistent, crisp taste that has garnered a significant following worldwide.

## II. Methodology

## 2.1 Data Understanding

**Products Table:** This table includes a variety of products that ACSE sold uniquely identified by a product ID. Each product has a description, category, sub-category, type, brand, and units of measure for a sale.

**Transactions Clean Table:** This table includes the transaction history from 6/24/2017 - 04/01/2020 with customer ID, store ID, product ID, transaction ID, sales quantity, weights, and amounts.

## 2.2 Data Preparation

**Identification of Frequent Beer Purchasers:** In the production table, we identified customers who purchased "domestic beer" from the subcategory at least three times and labeled them as frequent beer purchasers.

**Identification of aggregated sales data across various subcategories:** For frequent beer purchasers, we first computed the total sales amount for each product subcategory per customer. We then aggregated this sales data across numerous product subcategories such as 'Fresh Beef', 'Fresh Poultry', and others, calculating separate total sales amounts for each subcategory for every customer. We kept only the most important subcategories as determined by the Pareto principle based on the total sales amount (a total of 92 subcategories).

| cust_id | fresh_beef | fresh_poultry | deli_cheese | ready_to_eat | …… | alternative_beverage | mexican | dog_food |
|---|---|---|---|---|---|---|---|---|
| **1149745926** | 176.89 | 297.97 | 14.27 | 106.99 |  | 5 | 4.78 | 0 |

Table 1: Example Line

**Identification of highly correlated coefficients:** We conducted a correlation analysis on each column containing numerical values to assess the strength of relationships between variables. Upon examination, all relationships were found to have correlation coefficients less than 0.8. Consequently, no variables require removal from the analysis.

## 2.3 Clustering

We want to group similar frequent beer purchasers together in order to better predict the likelihood of customers in each cluster purchasing Molson Coors products.

Clustering is a machine learning technique used to group similar data points based on certain characteristics or features. We want to cluster the customers into subsets, where data points within the same cluster are more similar to each other than to those in other clusters. After we segment each customer into meaningful groups, we can use them to build logistic regression models that are more informative within each cluster instead of a single, generalized logistic regression model.

- *Standardization:* Features were standardized using the StandardScaler to ensure all variables contribute equally to the analysis, regardless of their original scale.
- *Determination of optimal number of clusters*: We use The Elbow Method to determine the optimal number of clusters for K-means clustering. This involved iterating over different numbers of clusters and choosing the smallest K where the rate of improvement in dispersion within clusters levels off.
- *K-means clustering:* Based on the optimal number of clusters determined from the Elbow Method, we perform the K-means clustering on the scaled features. In this scenario, we used three clusters.
- *Cluster analysis:* Each data point was assigned to one of the clusters based on its similarity in sub-category shopping behavior to the cluster centroids. The resulting clusters were analyzed to understand the characteristics of each cluster, such as which sub-categories of products outside of beer are most important.

**Results and Cluster Analysis:** From the Elbow Method, we find out the optimal number of clusters is three. Using the three clusters, we uncover the following insights:

- *Cluster 0 (Moderate Purchasing)*: This cluster represents average shoppers. They buy a mix of items in moderate amounts, indicating a balanced approach to purchases. These customers may be individuals or families with standard shopping habits and purchasing power that reflects typical consumer behavior.
- *Cluster 1 (Lower Purchasing)*: Customers in this cluster have the lowest average values across the categories, which suggests they are either more price-sensitive, purchase items less frequently, or buy in smaller quantities. They may have lower purchasing power or different shopping priorities compared to the other clusters.
- *Cluster 2 (Higher Purchasing):* This group has the highest values, suggesting that they have a higher purchasing volume across various categories. These customers likely have higher purchasing power or needs, which may be due to factors like larger household sizes, higher consumption, or a preference for stocking up. Also, they buy from more subcategories.

| Cluster | fresh_beef | fresh_poultry | deli_cheese | ready_to_eat | …… | mexican | dog_food |
|---|---|---|---|---|---|---|---|
| **0** | 336.17 | 347.12 | 300.22 | 174.87 | | 40.44 | 37.76 |
| **1** | 68.47 | 62.15 | 51.11 | 48.35 | | 7.56 | 7.42 |
| **2** | 797.36 | 864.13 | 781.61 | 317.25 | | 95.05 | 87.01 |

Table 2: Example of Clusters (Average)

## 2.4 Association Rules

Association rule mining finds patterns in data that can be represented as rules. These rules provide insights into the nature of relationships within the data, such as "If a customer buys item A, they are likely to buy item B." Association rules are measured by two key metrics:

- *Support*: The frequency with which items appear in the dataset.
- *Confidence*: The likelihood that the consequent (item B) is purchased when the antecedent (item A) is purchased.

We want to use association rules specific to each of the three clusters to identify which sub-categories can be considered antecedents for the consequent category "Domestic Beer". Later, the important sub-categories will be used as features in logistic regression models.

**Process:**

- *Clustered Thresholds for Binary Matrix:* We first convert the matrix to a binary matrix for frequent itemset processing. We set the threshold for cluster 0 to be 20, cluster 1 to be 0, and cluster 2 to be 20.
- *Cluster-Specific Thresholds:* When performing association rule mining on different clusters of data, it is important to set thresholds that are tailored to each unique characteristic of the cluster. For instance, clusters with generally higher transaction values (Cluster 0 and 2) require a higher support threshold to capture frequent, but not necessarily high-volume, associations. In contrast, for a cluster with lower overall transaction values (Cluster 1), a lower support threshold might be necessary to focus on the most significant patterns, as their overall number of transactions might be lower. As a result, we set 0.85 as support values for cluster 0, 0.92 for cluster 2, and 0.7 for cluster 1.
- *Customizing the Apriori Algorithm*: For each cluster, the Apriori algorithm's confidence parameter is adjusted. For example, a higher minimum confidence (like 0.9 or 0.95) is used for cluster 0 and cluster 2 where items are purchased together very frequently, while the same minimum confidence (0.9) is also applied for cluster 1, albeit with more varied purchase patterns.

**Results:** We found a list of antecedent sub-categories from the clusters that have an association with beer purchases. These sub-categories will be utilized as predictors in logistic regression analysis later on. In the final antecedent list, we have:

- *Antecedents in Cluster 0*: 'fresh_beef', 'fresh_poultry', 'deli_cheese', 'cheese', 'milk', 'root_veg', 'field_veg', 'salty_snacks', 'cooking_veg'
- *Antecedents in Cluster 1*: 'fresh_poultry', 'deli_cheese', 'ready_to_eat', 'cheese', 'milk', 'root_veg', 'field_veg', 'salty_snacks', 'cooking_veg', 'citrus', 'breads_commercial', 'tomatoes', 'salad_veg', 'eggs', 'bananas'
- *Antecedents in Cluster 2*: 'deli_cheese', 'cheese', 'berries_cherries', 'root_veg', 'field_veg', 'cooking_veg', 'baking_ingredients', 'citrus'

| | **Antecedents** | **Consequents** |
|---|---|---|
| **0** | **fresh_poultry** | **domestic_beer** |
| **1** | **deli_cheese** | **domestic_beer** |
| **2** | **ready_to_eat** | **domestic_beer** |
| | **…** | |
| **5** | **cheese** | **domestic_beer** |

Table 3: Example Result of Association Rule in Cluster 1

## 2.5 Logistic Regression

We opt for logistic regression instead of Naive Bayes due to Naive Bayes' assumption of independence among features. Our observation reveals co-subcategories in association rules, which suggests that the features are not independent, making logistic regression a more suitable choice. In logistic regression, we use each cluster's antecedent list as features to define the target variable as a binary outcome. We classify whether a customer purchases Molson Coors beer or not.

**Process:**

- *Preventing Data Leakage:* We remove Budweiser customers before fitting models to avoid data leakage.
- *Building for High-Engagement Products:* In order to identify the relevant products, we built logistic regression models for products where over 1% of the cluster population has purchased them.
- *Cluster-Specific Logistic Regression Models:* We use a set of logistic regression models for each cluster.

- *Model Training and Evaluation:* We split the dataset to the training and testing dataset with a 70:30 split. We standardize the feature values using StandardScaler to ensure consistency in scale. We then evaluate the accuracy of the model on the testing set to assess its performance.
- *Utilizing Predicted Probabilities:* Once every model in each cluster is trained, we employ it to predict probabilities for each sample in the testing set, yielding probability scores representing the likelihood of belonging to the positive class - the customer purchased Molson Coor product.

**Results:**

| Number of purchasers for 'bought_21183452: 2450 |
|---|
| ... |
| Number of purchasers for 'bought_20944740': 9737 |

Table 4: Example Result of Sum of Products in Cluster 1

| Accuracy for bought_21183452: 0.98 |
|---|
| ... |
| Accuracy for bought_20944740: 0.94 |

Table 5: Example Result of Accuracy Score of Logistic Regression in Cluster 1

| bought_21183452 | bought_21183451 | ... | bought_20944740 | cust_id |
|---|---|---|---|---|
| 0.022 | 0.024 | ... | 0.0087 | 1134016245 |

Table 6: Example Result of Prediction of Logistic Regression in Cluster 1

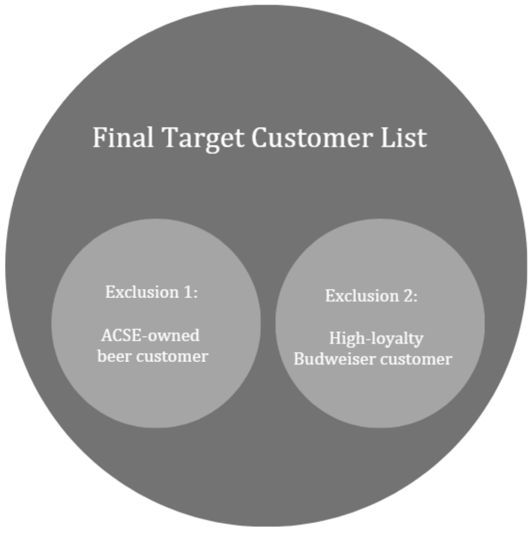
## III. Analysis

### 3.1 Target Customer List

The main objective is to promote Molson Coors products to Budweiser consumers via a recommender system without hurting ACSE through the partnership - we cannot recommend Molson Coors products to ACSE customers who have purchased beer. It is neither cost-efficient nor appropriate from a business standpoint to target all Budweiser customers, additional considerations are incorporated to refine the targeted customer list:

- *Minimizing potential churn of ACSE-owned beer customers*: exclude customers who have purchased ACSE-manufactured beer products
- *Avoiding targeting highly loyal Budweiser customers:* exclude the 20th percent of customers who only purchase Budweiser products within the 'Domestic Beer' subcategory.

The recommender system will not target those who purchased ACSE beer products or are loyal to Budweiser products. We exclude customers in the top 20th percent in terms of Budweiser product purchases, filtering for those who exclusively buy Budweiser within the 'Domestic Beer' category. The visualization of target customer list is shown below:



Graph 1: All Customers with Budweiser Purchasing History

The refinement of the targeted customer list identifies Budweiser customers who are open to other beer brands, while protecting ACSE's own customer group.

## 3.2 Predictions

Referring to Section 2.3, the list of target customers were fitted into clusters based on their expenditure within key subcategories. According to the cluster each customer is assigned, respective regression models are deployed to predict the likelihood of each Molson Coors product purchase, as outlined in Section 2.5.

For the purposes of making recommendations, the product with the highest probability of purchase for each customer is identified as the potential recommendation and is prioritized. A sample of the results table is presented below:

| cust_id | product_id | prod_desc | predicted_probability |
|---|---|---|---|
| 1132183813 | 20355000 | RICKARD'S RED BEER | 0.076579 |
| 1126070627 | 20946471 | COORS LIGHT 6 PK TC | 0.103740 |
| 1125913617 | 20811645 | COORS BANQUET BEER CAN 473ML | 0.067620 |
| 1152003806 | 21097267 | CREEMORE SPRINGS PREMIUM LAGER | 0.075184 |
| 1144713807 | 20946471 | COORS LIGHT 6 PK TC | 0.113453 |

Table 7: First 5 rows of the result table

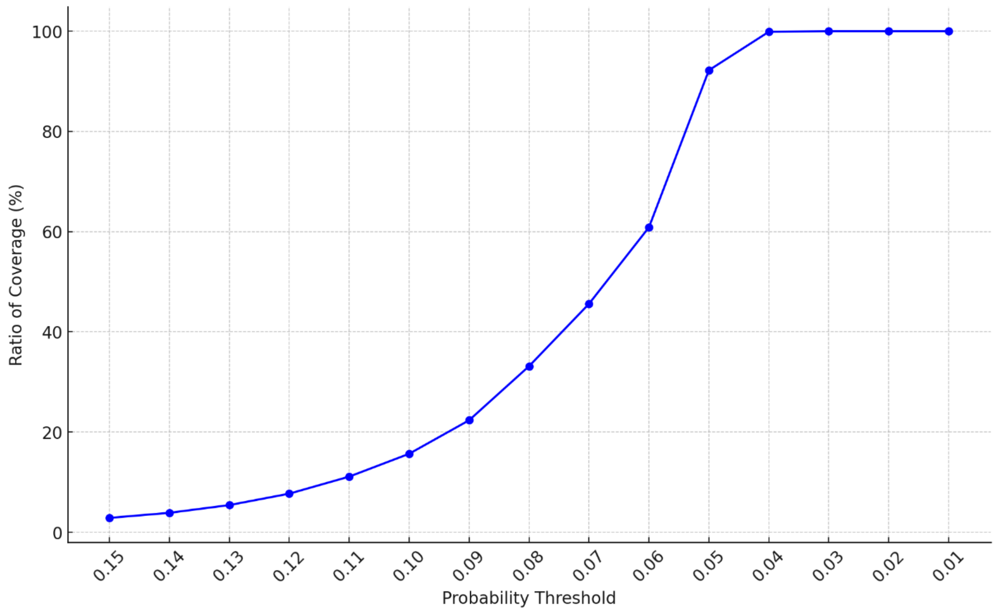
- *Cost-Oriented Threshold:* Although all targeted customers are assigned with specific Molson Coors product recommendation, it is worth noting that a probability threshold should be set in light of cost-benefit analysis. The result suggests a probability threshold of 0.10 to be financially optimal:

| | Value | Formula | Note |
|---|---|---|---|
| Benefit | $0.504 | Est. Profit Margin (4.8%) * Average Price of Popular Products ($10.50) | Popular Products include: - 20811645 (Price $4.24) - 20946471 (Price $13.28) - 20944740 (Price $13.98) |
| Cost | $0.05 | Est. dollar spent per recommendation offer | N/A |
| Break-even threshold | 0.10 | Cost / Benefit | N/A |

Table 8: Process table for computing financially preferred threshold

- *Strategy-Oriented Threshold:* Nevertheless, given Molson Coors' strategic objective to compete for Budweiser's customers, coverage of the recommendation is paramount. Our strategy-oriented threshold is computed by slacking 50% of the cost-oriented threshold, lowering from 0.10 to 0.05.

As illustrated in Graph 2 below, relaxing the threshold from 0.10 to 0.05 significantly enhances the recommendation coverage of the target customer group from 16% to 92%. With 0.05 threshold, which is a effective point for mass recommendations, the recommender system will cover 42,437 customers from Budweiser as shown in table 9:



Graph 2: Relationship between % Sent Recommendation and Probability Threshold

| Probability Threshold | Coverage Ratio (%) | Number of Recommended Customers |
|---|---|---|
| **0.05** | **92.20** | **42,437** |
| **0.06** | **60.86** | **28,010** |
| **0.07** | **45.59** | **20,984** |
| **0.08** | **33.15** | **15,256** |
| **0.09** | **22.35** | **10,288** |
| **0.10** | **15.66** | **7,209** |

Table 9: Display of Recommendation Offer Upon Different Probability Thresholds

# IV. Challenges and Conclusion

## 4.1 Challenges

- Challenge 1: Single Recommendation Per Customer

Due to cost-efficiency considerations, the approach of offering only one recommendation per customer (selecting the Molson Coors product with the highest purchasing probability) has limitations. As customers might have a likelihood to purchase multiple products, this strategy may not fully capitalize on potential customer interests, and result in missed opportunities for additional sales.

- Challenge 2: Customer Preference Dynamics

The fact that the current recommender system is developed upon the assumption that customer behavior is static. In real scenarios, customers' purchasing patterns may change overtime due to various unpredictable factors like shifting in general preference or influence from abrupt events (e.g., COVID-19). The recommender system should be constantly updated to account for this issue.

- Challenge 3: Impact on Inventory Management

Depending on the effectiveness of recommendations, the increased purchases of the top-recommended Molson Coors products could create disturbance in inventory management: maintaining sufficient stock levels to meet the boosted demand driven by the recommendation system is resource-intensive. Additional resources must be invested into inventory management to prevent overstock and understock situations, this will tie up capital and increase holding costs for ACSE supermarket.

- Challenge 4: Unspecific recommendation using subcategory

Currently, our recommendations are primarily based on subcategories rather than specific products. By relying on broader subcategory information, there is a potential risk of overlooking unique attributes and distinctions of each product which might not be sufficiently tailored to individual preferences or needs.

## 4.2 Conclusion

Our team aims to create a recommender system specifically tailored to boost the sales of Molson Coors products by strategically targeting Budweiser customers. Our methodology incorporated a combination of data preparation, clustering, association rule mining, and logistic regression. After adopting this method, we were able to segment customers into distinct clusters based on purchasing patterns, identify significant associations between product sub-categories, and predict the likelihood of switching to Molson Coors products with specific logistic regression models for each cluster and product.

We further refined our strategy by excluding customers who purchase ACSE-branded beer and customers highly loyal to Budweiser. We also determined the probability threshold through our cost-benefit analysis with two threshold options. We chose the lower threshold to achieve broader customer reach, prioritizing market penetration over immediate financial return. We believe this threshold aligns with Molson Coors' competitive approach. In conclusion, our in-depth analysis has assisted Molson Coors by identifying the best Budweiser customers to recommend specific products to, and this will lead to increased sales and higher customer satisfaction by offering a more personalized shopping experience.

## Codebook Content

**SQL (tool : pgadmin & bigquery)** :

1: acse_beer_customers
2: did_buy_mc_products
3: final relevant products list
4: customers who have bought budweiser no non alcoholic
5: beer_customer_subcat
6: target_cust_id

**Python (tool: google colab):**

Clustering

7: cluster_0_ids
8: cluster_1_ids
9: cluster_2_ids

Association Rule on Each Cluster

Logistic Regression on Each Cluster

Final Prediction

**Table 1:**

```
SELECT DISTINCT tc.cust_id
FROM transactions_clean tc
INNER JOIN products p ON tc.prod_id = p.prod_id
WHERE p.prod_mfc_brand_cd = 'ACSE'
AND (p.prod_category LIKE '%Beer%' OR p.prod_subcategory LIKE '%Beer%');
```

**Table 2:**

```
SELECT
  t.cust_id,
  MAX(CASE WHEN t.prod_id = '20975816' THEN 1 ELSE 0 END) AS bought_20975816,
  MAX(CASE WHEN t.prod_id = '20975814' THEN 1 ELSE 0 END) AS bought_20975814,
  MAX(CASE WHEN t.prod_id = '20838127' THEN 1 ELSE 0 END) AS bought_20838127,
  MAX(CASE WHEN t.prod_id = '21182258' THEN 1 ELSE 0 END) AS bought_21182258,
  MAX(CASE WHEN t.prod_id = '21279935' THEN 1 ELSE 0 END) AS bought_21279935,
  MAX(CASE WHEN t.prod_id = '21279941' THEN 1 ELSE 0 END) AS bought_21279941,
  MAX(CASE WHEN t.prod_id = '21291851' THEN 1 ELSE 0 END) AS bought_21291851,
  MAX(CASE WHEN t.prod_id = '21294687' THEN 1 ELSE 0 END) AS bought_21294687,
  MAX(CASE WHEN t.prod_id = '21296693' THEN 1 ELSE 0 END) AS bought_21296693,
  MAX(CASE WHEN t.prod_id = '21354895' THEN 1 ELSE 0 END) AS bought_21354895,
  MAX(CASE WHEN t.prod_id = '20880905' THEN 1 ELSE 0 END) AS bought_20880905,
  MAX(CASE WHEN t.prod_id = '20955226' THEN 1 ELSE 0 END) AS bought_20955226,
  MAX(CASE WHEN t.prod_id = '21098750' THEN 1 ELSE 0 END) AS bought_21098750,
  MAX(CASE WHEN t.prod_id = '21189291' THEN 1 ELSE 0 END) AS bought_21189291,
  MAX(CASE WHEN t.prod_id = '21218683' THEN 1 ELSE 0 END) AS bought_21218683,
  MAX(CASE WHEN t.prod_id = '20589944' THEN 1 ELSE 0 END) AS bought_20589944,
  MAX(CASE WHEN t.prod_id = '20105629' THEN 1 ELSE 0 END) AS bought_20105629,
  MAX(CASE WHEN t.prod_id = '20712548' THEN 1 ELSE 0 END) AS bought_20712548,
  MAX(CASE WHEN t.prod_id = '21190030' THEN 1 ELSE 0 END) AS bought_21190030,
  MAX(CASE WHEN t.prod_id = '21097262' THEN 1 ELSE 0 END) AS bought_21097262,
  MAX(CASE WHEN t.prod_id = '21193266' THEN 1 ELSE 0 END) AS bought_21193266,
  MAX(CASE WHEN t.prod_id = '21217297' THEN 1 ELSE 0 END) AS bought_21217297,
  MAX(CASE WHEN t.prod_id = '21183454' THEN 1 ELSE 0 END) AS bought_21183454,
  MAX(CASE WHEN t.prod_id = '21185185' THEN 1 ELSE 0 END) AS bought_21185185,
  MAX(CASE WHEN t.prod_id = '21204179' THEN 1 ELSE 0 END) AS bought_21204179,
  MAX(CASE WHEN t.prod_id = '21183452' THEN 1 ELSE 0 END) AS bought_21183452,
  MAX(CASE WHEN t.prod_id = '21183451' THEN 1 ELSE 0 END) AS bought_21183451,
  MAX(CASE WHEN t.prod_id = '21101242' THEN 1 ELSE 0 END) AS bought_21101242,
  MAX(CASE WHEN t.prod_id = '20014425' THEN 1 ELSE 0 END) AS bought_20014425,
  MAX(CASE WHEN t.prod_id = '21185186' THEN 1 ELSE 0 END) AS bought_21185186,
  MAX(CASE WHEN t.prod_id = '20165323' THEN 1 ELSE 0 END) AS bought_20165323,
  MAX(CASE WHEN t.prod_id = '21097267' THEN 1 ELSE 0 END) AS bought_21097267,
  MAX(CASE WHEN t.prod_id = '20887682' THEN 1 ELSE 0 END) AS bought_20887682,
  MAX(CASE WHEN t.prod_id = '20563488' THEN 1 ELSE 0 END) AS bought_20563488,
  MAX(CASE WHEN t.prod_id = '20954763' THEN 1 ELSE 0 END) AS bought_20954763,
  MAX(CASE WHEN t.prod_id = '21094401' THEN 1 ELSE 0 END) AS bought_21094401,
  MAX(CASE WHEN t.prod_id = '20584739' THEN 1 ELSE 0 END) AS bought_20584739,
  MAX(CASE WHEN t.prod_id = '21098749' THEN 1 ELSE 0 END) AS bought_21098749,
  MAX(CASE WHEN t.prod_id = '20811645' THEN 1 ELSE 0 END) AS bought_20811645,
  MAX(CASE WHEN t.prod_id = '20355000' THEN 1 ELSE 0 END) AS bought_20355000,
  MAX(CASE WHEN t.prod_id = '20946471' THEN 1 ELSE 0 END) AS bought_20946471,
  MAX(CASE WHEN t.prod_id = '20990264' THEN 1 ELSE 0 END) AS bought_20990264,
  MAX(CASE WHEN t.prod_id = '20944740' THEN 1 ELSE 0 END) AS bought_20944740,
  MAX(CASE WHEN t.prod_id = '20000192' THEN 1 ELSE 0 END) AS bought_20000192,
  MAX(CASE WHEN t.prod_id = '20954757' THEN 1 ELSE 0 END) AS bought_20954757,
  MAX(CASE WHEN t.prod_id = '20079283' THEN 1 ELSE 0 END) AS bought_20079283,
  MAX(CASE WHEN t.prod_id = '21103816' THEN 1 ELSE 0 END) AS bought_21103816,
  MAX(CASE WHEN t.prod_id = '21103977' THEN 1 ELSE 0 END) AS bought_21103977,
  MAX(CASE WHEN t.prod_id = '20373179' THEN 1 ELSE 0 END) AS bought_20373179,
  MAX(CASE WHEN t.prod_id = '21105236' THEN 1 ELSE 0 END) AS bought_21105236,
  MAX(CASE WHEN t.prod_id = '20056727' THEN 1 ELSE 0 END) AS bought_20056727
FROM
  transactions_clean t
INNER JOIN
  products p ON t.prod_id = p.prod_id
WHERE
  p.prod_subcategory = 'Domestic Beer'
  AND t.cust_id IN (
      SELECT
          t.cust_id
      FROM
          transactions_clean t
      INNER JOIN
          products p ON t.prod_id = p.prod_id
      WHERE
          p.prod_subcategory = 'Domestic Beer'
      GROUP BY
          t.cust_id
      HAVING
          COUNT(DISTINCT t.prod_id) >= 3
  )
GROUP BY
  t.cust_id;
```

**Table 3:**

```
SELECT *,
    CASE
        WHEN LOWER(prod_desc) LIKE '%alexander keith''s%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%appalachian mountain brewery%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%blue star%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%boomerang%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%bud%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%budweiser%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%busch%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%cisco brewers%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%devils backbone brewing company%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%double deer%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%eagle lager%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%golden road brewing%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%goose island brewery%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%hell''s gate%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%herman joseph''s private reserve%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%malta morena%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%michelob%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%michelob ultra%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%mill street brewery%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%modelo especial%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%oland export ale%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%omission brewery%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%presidente%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%platform beer co.%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%rolling rock%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%ron barceló%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%schooner lager%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%stanley park%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%turning point%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%veza sur brewing co%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%wicked weed%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%wynwood brewing%' THEN 'Ab Inbev'
        WHEN LOWER(prod_desc) LIKE '%barmen%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%black horse%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%black ice%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%blue moon%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%bohemian%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%colorado native%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%coors%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%creemore springs%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%crispin cider%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%dundee brewing%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%extra gold lager%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%george killian''s irish red%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%granville island%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%hamm''s%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%henry weinhard''s%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%henry''s soda%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%hop valley brewing%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%icehouse%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%india beer%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%keystone%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%laurentide%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%leinenkugel''s%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%mad and noisy%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%mad jack cider%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%magnum%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%mickey''s%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%miller%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%milwaukee''s best%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%molson%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%old style pilsner%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%old vienna%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%olde english%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%red dog%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%revolver brewing%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%rickard''s%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%saint archer%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%smith and forge cider%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%sparks%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%standard lager%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%staropramen%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%steel reserve%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%terrapin%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%third shift%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%wanderoot%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%winterfest%' THEN 'Molson Coors'
        WHEN LOWER(prod_desc) LIKE '%zumbida mango%' THEN 'Molson Coors'
        ELSE NULL
    END AS company
```

```
FROM products
WHERE (prod_category LIKE '%Beer%' OR prod_subcategory LIKE '%Beer%') AND (
                    LOWER(prod_desc) LIKE '%alexander keith''s%'
    OR LOWER(prod_desc) LIKE '%appalachian mountain brewery%'
    OR LOWER(prod_desc) LIKE '%blue star%'
    OR LOWER(prod_desc) LIKE '%boomerang%'
    OR LOWER(prod_desc) LIKE '%bud%'
    OR LOWER(prod_desc) LIKE '%budweiser%'
    OR LOWER(prod_desc) LIKE '%busch%'
    OR LOWER(prod_desc) LIKE '%cisco brewers%'
    OR LOWER(prod_desc) LIKE '%devils backbone brewing company%'
    OR LOWER(prod_desc) LIKE '%double deer%'
    OR LOWER(prod_desc) LIKE '%eagle lager%'
    OR LOWER(prod_desc) LIKE '%golden road brewing%'
    OR LOWER(prod_desc) LIKE '%goose island brewery%'
    OR LOWER(prod_desc) LIKE '%hell''s gate%'
    OR LOWER(prod_desc) LIKE '%herman joseph''s private reserve%'
    OR LOWER(prod_desc) LIKE '%malta morena%'
    OR LOWER(prod_desc) LIKE '%michelob%'
    OR LOWER(prod_desc) LIKE '%michelob ultra%'
    OR LOWER(prod_desc) LIKE '%mill street brewery%'
    OR LOWER(prod_desc) LIKE '%modelo especial%'
    OR LOWER(prod_desc) LIKE '%oland export ale%'
    OR LOWER(prod_desc) LIKE '%omission brewery%'
    OR LOWER(prod_desc) LIKE '%presidente%'
    OR LOWER(prod_desc) LIKE '%platform beer co.%'
    OR LOWER(prod_desc) LIKE '%rolling rock%'
    OR LOWER(prod_desc) LIKE '%ron barceló%'
    OR LOWER(prod_desc) LIKE '%schooner lager%'
    OR LOWER(prod_desc) LIKE '%stanley park%'
    OR LOWER(prod_desc) LIKE '%turning point%'
    OR LOWER(prod_desc) LIKE '%veza sur brewing co%'
    OR LOWER(prod_desc) LIKE '%wicked weed%'
    OR LOWER(prod_desc) LIKE '%wynwood brewing%'
    OR LOWER(prod_desc) LIKE '%barmen%'
    OR LOWER(prod_desc) LIKE '%black horse%'
    OR LOWER(prod_desc) LIKE '%black ice%'
    OR LOWER(prod_desc) LIKE '%blue moon%'
    OR LOWER(prod_desc) LIKE '%bohemian%'
    OR LOWER(prod_desc) LIKE '%colorado native%'
    OR LOWER(prod_desc) LIKE '%coors%'
    OR LOWER(prod_desc) LIKE '%creemore springs%'
    OR LOWER(prod_desc) LIKE '%crispin cider%'
    OR LOWER(prod_desc) LIKE '%dundee brewing%'
    OR LOWER(prod_desc) LIKE '%extra gold lager%'
    OR LOWER(prod_desc) LIKE '%george killian''s irish red%'
    OR LOWER(prod_desc) LIKE '%granville island%'
    OR LOWER(prod_desc) LIKE '%hamm''s%'
    OR LOWER(prod_desc) LIKE '%henry weinhard''s%'
    OR LOWER(prod_desc) LIKE '%henry''s soda%'
    OR LOWER(prod_desc) LIKE '%hop valley brewing%'
    OR LOWER(prod_desc) LIKE '%icehouse%'
    OR LOWER(prod_desc) LIKE '%india beer%'
    OR LOWER(prod_desc) LIKE '%keystone%'
    OR LOWER(prod_desc) LIKE '%laurentide%'
    OR LOWER(prod_desc) LIKE '%leinenkugel''s%'
    OR LOWER(prod_desc) LIKE '%mad and noisy%'
    OR LOWER(prod_desc) LIKE '%mad jack cider%'
    OR LOWER(prod_desc) LIKE '%magnum%'
    OR LOWER(prod_desc) LIKE '%mickey''s%'
    OR LOWER(prod_desc) LIKE '%miller%'
    OR LOWER(prod_desc) LIKE '%milwaukee''s best%'
    OR LOWER(prod_desc) LIKE '%molson%'
    OR LOWER(prod_desc) LIKE '%old style pilsner%'
    OR LOWER(prod_desc) LIKE '%old vienna%'
    OR LOWER(prod_desc) LIKE '%olde english%'
    OR LOWER(prod_desc) LIKE '%red dog%'
    OR LOWER(prod_desc) LIKE '%revolver brewing%'
    OR LOWER(prod_desc) LIKE '%rickard''s%'
    OR LOWER(prod_desc) LIKE '%saint archer%'
    OR LOWER(prod_desc) LIKE '%smith and forge cider%'
    OR LOWER(prod_desc) LIKE '%sparks%'
    OR LOWER(prod_desc) LIKE '%standard lager%'
    OR LOWER(prod_desc) LIKE '%staropramen%'
    OR LOWER(prod_desc) LIKE '%steel reserve%'
    OR LOWER(prod_desc) LIKE '%terrapin%'
    OR LOWER(prod_desc) LIKE '%third shift%'
    OR LOWER(prod_desc) LIKE '%wanderoot%'
    OR LOWER(prod_desc) LIKE '%winterfest%'
    OR LOWER(prod_desc) LIKE '%zumbida mango%')
;

SELECT prod_id, COUNT(trans_id) AS transaction_count, COUNT(DISTINCT cust_id) AS
unique_cust_count
FROM transactions_clean
WHERE prod_id IN (
    20056727, 21228044, 21295370, 21295398, 21299616,
    20079283, 20373179, 20970847, 20971422, 20971430,
    20971436, 20971437, 21103816, 21103977, 21105236,
    20014425, 21094401, 21095737, 20989759, 20990264,
    21097262, 21097267, 21338229, 20140413, 21098749,
    21098750, 20355000, 20975814, 20975816, 21101242,
    21296693, 20505065, 20505066, 20712548, 21279935,
    21279941, 21280029, 21303373, 21290794, 21291851,
    21294687, 21354895, 20060065, 20105629, 20955258,
    20115763, 21192047, 20165323, 20181157, 20563488,
    21218683, 21200711, 21193248, 20584739, 20589944,
    21193266, 21182258, 21042403, 21042691, 21217297,
    21183451, 21183452, 21183454, 21185185, 20944739,
    20944740, 21185186, 20946471, 20000192, 20811645,
    20838127, 21189291, 21190030, 21190075, 21204179,
    20954757, 20954763, 20954800, 20880905, 20887682,
    20955226
)
GROUP BY prod_id
HAVING COUNT(cust_id) > 0;
```

**Table 4:**

```
SELECT tc.
FROM transactions_clean tc
JOIN (
    SELECT DISTINCT cust_id
    FROM transactions_clean
    WHERE prod_id IN (
        20060065, 20115763, 20140413, 20181157, 20505065,
        20505066, 20944739, 20954800, 20955258, 20989759,
        21042403, 21042691, 21095737, 21190075, 21192047,
        21193248, 21200711, 21280029, 21290794, 21303373,
        21338229
    ) AND cust_id IS NOT NULL
) t ON tc.cust_id = t.cust_id;
```

**Table 5:**

```
WITH BeerPurchasers AS (
    SELECT
```

```
        t.cust_id
    FROM transactions_clean t
    INNER JOIN products p ON t.prod_id = p.prod_id
    WHERE p.prod_subcategory = 'Domestic Beer'
    GROUP BY t.cust_id
    HAVING COUNT(DISTINCT t.prod_id) >= 3
),
CategorySales AS (
    SELECT
        t.cust_id,
        p.prod_subcategory,
        SUM(t.sales_amt) AS total_sales_amt
    FROM transactions_clean t
    INNER JOIN products p ON t.prod_id = p.prod_id
    INNER JOIN BeerPurchasers bp ON t.cust_id = bp.cust_id
    GROUP BY t.cust_id, p.prod_subcategory
)
SELECT
    cs.cust_id,
    SUM(CASE WHEN cs.prod_subcategory = 'Fresh-Beef' THEN cs.total_sales_amt ELSE 0 END) AS
Fresh_Beef,
    SUM(CASE WHEN cs.prod_subcategory = 'Fresh-Poultry' THEN cs.total_sales_amt ELSE 0 END) AS
Fresh_Poultry,
    SUM(CASE WHEN cs.prod_subcategory = 'Deli Cheese' THEN cs.total_sales_amt ELSE 0 END) AS
Deli_Cheese,
    SUM(CASE WHEN cs.prod_subcategory = 'Ready to Eat' THEN cs.total_sales_amt ELSE 0 END) AS
Ready_to_Eat,
    SUM(CASE WHEN cs.prod_subcategory = 'Cheese' THEN cs.total_sales_amt ELSE 0 END) AS
Cheese,
    SUM(CASE WHEN cs.prod_subcategory = 'Milk' THEN cs.total_sales_amt ELSE 0 END) AS Milk,
    SUM(CASE WHEN cs.prod_subcategory = 'Berries/Cherries' THEN cs.total_sales_amt ELSE 0 END)
AS Berries_Cherries,
    SUM(CASE WHEN cs.prod_subcategory = 'Root Veg' THEN cs.total_sales_amt ELSE 0 END) AS
Root_Veg,
    SUM(CASE WHEN cs.prod_subcategory = 'Sweets-In-Store' THEN cs.total_sales_amt ELSE 0 END)
AS Sweets_In_Store,
    SUM(CASE WHEN cs.prod_subcategory = 'Yogurt' THEN cs.total_sales_amt ELSE 0 END) AS Yogurt,
    SUM(CASE WHEN cs.prod_subcategory = 'Field Veg' THEN cs.total_sales_amt ELSE 0 END) AS
Field_Veg,
    SUM(CASE WHEN cs.prod_subcategory = 'Salty Snacks' THEN cs.total_sales_amt ELSE 0 END) AS
Salty_Snacks,
    SUM(CASE WHEN cs.prod_subcategory = 'Packaged Salads' THEN cs.total_sales_amt ELSE 0 END)
AS Packaged_Salads,
   SUM(CASE WHEN cs.prod_subcategory = 'Cooking Veg' THEN cs.total_sales_amt ELSE 0 END) AS
Cooking_Veg,
   SUM(CASE WHEN cs.prod_subcategory = 'Self Serve Deli Meat' THEN cs.total_sales_amt ELSE 0
END) AS Self_Serve_Deli_Meat,
   SUM(CASE WHEN cs.prod_subcategory = 'Coffee' THEN cs.total_sales_amt ELSE 0 END) AS Coffee,
   SUM(CASE WHEN cs.prod_subcategory = 'Gourmet Foods' THEN cs.total_sales_amt ELSE 0 END)
AS Gourmet_Foods,
   SUM(CASE WHEN cs.prod_subcategory = 'Baking Ingredients' THEN cs.total_sales_amt ELSE 0
END) AS Baking_Ingredients,
   SUM(CASE WHEN cs.prod_subcategory = 'Dairy-Natural Foods' THEN cs.total_sales_amt ELSE 0
END) AS Dairy_Natural_Foods,
   SUM(CASE WHEN cs.prod_subcategory = 'Citrus' THEN cs.total_sales_amt ELSE 0 END) AS Citrus,
   SUM(CASE WHEN cs.prod_subcategory = 'Domestic Beer' THEN cs.total_sales_amt ELSE 0 END)
AS Domestic_Beer,
   SUM(CASE WHEN cs.prod_subcategory = 'Breads-Commercial' THEN cs.total_sales_amt ELSE 0
END) AS Breads_Commercial,
   SUM(CASE WHEN cs.prod_subcategory = 'Ice Cream' THEN cs.total_sales_amt ELSE 0 END) AS
Ice_Cream,
   SUM(CASE WHEN cs.prod_subcategory = 'Tomatoes' THEN cs.total_sales_amt ELSE 0 END) AS
Tomatoes,
   SUM(CASE WHEN cs.prod_subcategory = 'Apples' THEN cs.total_sales_amt ELSE 0 END) AS
Apples,
   SUM(CASE WHEN cs.prod_subcategory = 'Service Deli Meat' THEN cs.total_sales_amt ELSE 0 END)
AS Service_Deli_Meat,
   SUM(CASE WHEN cs.prod_subcategory = 'Fresh-Pork' THEN cs.total_sales_amt ELSE 0 END) AS
Fresh_Pork,
   SUM(CASE WHEN cs.prod_subcategory = 'Salad Veg' THEN cs.total_sales_amt ELSE 0 END) AS
Salad_Veg,
   SUM(CASE WHEN cs.prod_subcategory = 'Carbonated Soft Drin' THEN cs.total_sales_amt ELSE 0
END) AS Carbonated_Soft_Drin,
   SUM(CASE WHEN cs.prod_subcategory = 'Eggs' THEN cs.total_sales_amt ELSE 0 END) AS Eggs,
   SUM(CASE WHEN cs.prod_subcategory = 'Cereal Rte' THEN cs.total_sales_amt ELSE 0 END) AS
Cereal_Rte,
   SUM(CASE WHEN cs.prod_subcategory = 'Water' THEN cs.total_sales_amt ELSE 0 END) AS Water,
   SUM(CASE WHEN cs.prod_subcategory = 'Tropical' THEN cs.total_sales_amt ELSE 0 END) AS
Tropical,
   SUM(CASE WHEN cs.prod_subcategory = 'Crackers/Health Cake' THEN cs.total_sales_amt ELSE 0
END) AS Crackers_Health_Cake,
   SUM(CASE WHEN cs.prod_subcategory = 'Fresh Salmon' THEN cs.total_sales_amt ELSE 0 END) AS
Fresh_Salmon,
   SUM(CASE WHEN cs.prod_subcategory = 'Bathroom Tissue' THEN cs.total_sales_amt ELSE 0 END)
AS Bathroom_Tissue,
   SUM(CASE WHEN cs.prod_subcategory = 'Peppers' THEN cs.total_sales_amt ELSE 0 END) AS
Peppers,
   SUM(CASE WHEN cs.prod_subcategory = 'Cigarettes' THEN cs.total_sales_amt ELSE 0 END) AS
Cigarettes,
   SUM(CASE WHEN cs.prod_subcategory = 'Butter & Margarine' THEN cs.total_sales_amt ELSE 0
END) AS Butter_Margarine,
   SUM(CASE WHEN cs.prod_subcategory = 'Sauces/Marinades' THEN cs.total_sales_amt ELSE 0
END) AS Sauces_Marinades,
   SUM(CASE WHEN cs.prod_subcategory = 'Grapes' THEN cs.total_sales_amt ELSE 0 END) AS
Grapes,
   SUM(CASE WHEN cs.prod_subcategory = 'Alternatives-Commerc' THEN cs.total_sales_amt ELSE 0
END) AS Alternatives_Commerc,
   SUM(CASE WHEN cs.prod_subcategory = 'Breads-In-Store' THEN cs.total_sales_amt ELSE 0 END)
AS Breads_In_Store,
   SUM(CASE WHEN cs.prod_subcategory = 'Laundry-Household Cl' THEN cs.total_sales_amt ELSE 0
END) AS Laundry_Household_Cl,
   SUM(CASE WHEN cs.prod_subcategory = 'Chocolate' THEN cs.total_sales_amt ELSE 0 END) AS
Chocolate,
   SUM(CASE WHEN cs.prod_subcategory = 'Juices & Drinks-Refr' THEN cs.total_sales_amt ELSE 0
END) AS Juices_Drinks_Refr,
   SUM(CASE WHEN cs.prod_subcategory = 'Domestic Wine' THEN cs.total_sales_amt ELSE 0 END)
AS Domestic_Wine,
   SUM(CASE WHEN cs.prod_subcategory = 'Confectionary / Bars' THEN cs.total_sales_amt ELSE 0
END) AS Confectionary_Bars,
   SUM(CASE WHEN cs.prod_subcategory = 'Bacon' THEN cs.total_sales_amt ELSE 0 END) AS Bacon,
   SUM(CASE WHEN cs.prod_subcategory = 'Dinner & Entres' THEN cs.total_sales_amt ELSE 0 END)
AS Dinner_Entres,
   SUM(CASE WHEN cs.prod_subcategory = 'Snacks-Natural Foods' THEN cs.total_sales_amt ELSE 0
END) AS Snacks_Natural_Foods,
   SUM(CASE WHEN cs.prod_subcategory = 'Side Dish/Rice' THEN cs.total_sales_amt ELSE 0 END)
AS Side_Dish_Rice,
   SUM(CASE WHEN cs.prod_subcategory = 'Frozen-Natural Foods' THEN cs.total_sales_amt ELSE 0
END) AS Frozen_Natural_Foods,
   SUM(CASE WHEN cs.prod_subcategory = 'Sushi' THEN cs.total_sales_amt ELSE 0 END) AS Sushi,
   SUM(CASE WHEN cs.prod_subcategory = 'Ready To Heat' THEN cs.total_sales_amt ELSE 0 END) AS
Ready_To_Heat,
   SUM(CASE WHEN cs.prod_subcategory = 'Rolls-In-Store' THEN cs.total_sales_amt ELSE 0 END) AS
Rolls_In_Store,
   SUM(CASE WHEN cs.prod_subcategory = 'Cut Fruit' THEN cs.total_sales_amt ELSE 0 END) AS
Cut_Fruit,
   SUM(CASE WHEN cs.prod_subcategory = 'Bananas' THEN cs.total_sales_amt ELSE 0 END) AS
Bananas,
  SUM(CASE WHEN cs.prod_subcategory = 'Cookies' THEN cs.total_sales_amt ELSE 0 END) AS
Cookies,
  SUM(CASE WHEN cs.prod_subcategory = 'Cut Flowers' THEN cs.total_sales_amt ELSE 0 END) AS
Cut_Flowers,
  SUM(CASE WHEN cs.prod_subcategory = 'Juices & Drinks' THEN cs.total_sales_amt ELSE 0 END)
AS Juices_Drinks,
  SUM(CASE WHEN cs.prod_subcategory = 'Spreads' THEN cs.total_sales_amt ELSE 0 END) AS
Spreads,
  SUM(CASE WHEN cs.prod_subcategory = 'Cut Veg' THEN cs.total_sales_amt ELSE 0 END) AS
Cut_Veg,
  SUM(CASE WHEN cs.prod_subcategory = 'Breakfast-Natural Fo' THEN cs.total_sales_amt ELSE 0
END) AS Breakfast_Natural_Fo,
  SUM(CASE WHEN cs.prod_subcategory = 'Canned Soup' THEN cs.total_sales_amt ELSE 0 END) AS
Canned_Soup,
  SUM(CASE WHEN cs.prod_subcategory = 'Shrimp' THEN cs.total_sales_amt ELSE 0 END) AS
Shrimp,
  SUM(CASE WHEN cs.prod_subcategory = 'Baking-Spices' THEN cs.total_sales_amt ELSE 0 END) AS
Baking_Spices,
  SUM(CASE WHEN cs.prod_subcategory = 'Food Wrap & Bags' THEN cs.total_sales_amt ELSE 0
END) AS Food_Wrap_Bags,
  SUM(CASE WHEN cs.prod_subcategory = 'Pizza' THEN cs.total_sales_amt ELSE 0 END) AS Pizza,
  SUM(CASE WHEN cs.prod_subcategory = 'Nuts/Seeds/Snacks' THEN cs.total_sales_amt ELSE 0
END) AS Nuts_Seeds_Snacks,
  SUM(CASE WHEN cs.prod_subcategory = 'Baked Protein' THEN cs.total_sales_amt ELSE 0 END) AS
Baked_Protein,
  SUM(CASE WHEN cs.prod_subcategory = 'Fruit' THEN cs.total_sales_amt ELSE 0 END) AS Fruit,
  SUM(CASE WHEN cs.prod_subcategory = 'Condiments' THEN cs.total_sales_amt ELSE 0 END) AS
Condiments,
  SUM(CASE WHEN cs.prod_subcategory = 'Nutritional Portable' THEN cs.total_sales_amt ELSE 0
END) AS Nutritional_Portable,
  SUM(CASE WHEN cs.prod_subcategory = 'Oils' THEN cs.total_sales_amt ELSE 0 END) AS Oils,
  SUM(CASE WHEN cs.prod_subcategory = 'Canned Fish/Meat' THEN cs.total_sales_amt ELSE 0
END) AS Canned_Fish_Meat,
  SUM(CASE WHEN cs.prod_subcategory = 'Live Plant Material' THEN cs.total_sales_amt ELSE 0
END) AS Live_Plant_Material,
  SUM(CASE WHEN cs.prod_subcategory = 'Beverages-Natural Fo' THEN cs.total_sales_amt ELSE 0
END) AS Beverages_Natural_Fo,
  SUM(CASE WHEN cs.prod_subcategory = 'Vegetables-Frozen' THEN cs.total_sales_amt ELSE 0
END) AS Vegetables_Frozen,
  SUM(CASE WHEN cs.prod_subcategory = 'Sausage' THEN cs.total_sales_amt ELSE 0 END) AS
Sausage,
  SUM(CASE WHEN cs.prod_subcategory = 'Baking / Bulk-Natura' THEN cs.total_sales_amt ELSE 0
END) AS Baking_Bulk_Natura,
   SUM(CASE WHEN cs.prod_subcategory = 'Dressing/Dips/Juices' THEN cs.total_sales_amt ELSE 0
END) AS Dressing_Dips_Juices,
   SUM(CASE WHEN cs.prod_subcategory = 'Pasta' THEN cs.total_sales_amt ELSE 0 END) AS Pasta,
   SUM(CASE WHEN cs.prod_subcategory = 'Canned Vegetables' THEN cs.total_sales_amt ELSE 0
END) AS Canned_Vegetables,
   SUM(CASE WHEN cs.prod_subcategory = 'Meal Makers-Natural' THEN cs.total_sales_amt ELSE 0
END) AS Meal_Makers_Natural,
   SUM(CASE WHEN cs.prod_subcategory = 'Towels - Paper & Reu' THEN cs.total_sales_amt ELSE 0
END) AS Towels_Paper_Reu,
   SUM(CASE WHEN cs.prod_subcategory = 'Salad Dressings' THEN cs.total_sales_amt ELSE 0 END)
AS Salad_Dressings,
   SUM(CASE WHEN cs.prod_subcategory = 'Milk - Natural Foods' THEN cs.total_sales_amt ELSE 0
END) AS Milk_Natural_Foods,
   SUM(CASE WHEN cs.prod_subcategory = 'Disposable Diapers' THEN cs.total_sales_amt ELSE 0
END) AS Disposable_Diapers,
   SUM(CASE WHEN cs.prod_subcategory = 'Alternative Beverage' THEN cs.total_sales_amt ELSE 0
END) AS Alternative_Beverage,
   SUM(CASE WHEN cs.prod_subcategory = 'Mexican' THEN cs.total_sales_amt ELSE 0 END) AS
Mexican,
   SUM(CASE WHEN cs.prod_subcategory = 'Dog Food' THEN cs.total_sales_amt ELSE 0 END) AS
Dog_Food
FROM CategorySales cs
GROUP BY cs.cust_id;
```

**Table 6:**

```
QUERY = """
WITH BUD_Transactions AS (
  SELECT
    t.cust_id
  FROM  machine_learning.transactions_clean  t
  JOIN  machine_learning.products  p ON t.prod_id = p.prod_id
  WHERE p.prod_subcategory = 'Domestic Beer'
    AND p.prod_mfc_brand_cd IN ('BUDL', 'BUDW', 'UNBR')
  GROUP BY t.cust_id
  HAVING COUNT(DISTINCT t.trans_id) = COUNT(t.trans_id)
),
Transaction_Counts AS (
  SELECT
    t.cust_id,
    COUNT(*) as Transaction_Count
  FROM  machine_learning.transactions  t
  JOIN BUD_Transactions b ON t.cust_id = b.cust_id
  WHERE t.prod_id IN (SELECT prod_id FROM  machine_learning.products  WHERE
prod_mfc_brand_cd IN ('BUDL', 'BUDW', 'UNBR'))
  GROUP BY t.cust_id
),
Percentile_Calc AS (
  SELECT
    cust_id,
    Transaction_Count,
    PERCENT_RANK() OVER (ORDER BY Transaction_Count DESC) AS percentile
  FROM Transaction_Counts
)
SELECT cust_id
FROM Percentile_Calc
WHERE percentile <= 0.20;
"""

# Execute the query
query_job = client.query(QUERY)
results = query_job.result()

# Collect the results and save to a CSV file
final_cust_ids = [row.cust_id for row in results]
df_final = pd.DataFrame(final_cust_ids, columns=['cust_id'])
df_final.to_csv('/content/final_cust_ids.csv', index=False)
```

**Table 7:**

```
cluster_label = 0

# Filter the DataFrame to get the data for the specified cluster label
cluster_data = new_df[new_df['cluster'] == cluster_label]

# Assuming 'cust_id' is the column name for customer IDs
customer_ids = cluster_data['cust_id']

# Print customer IDs
print(customer_ids)

# Save to CSV file
filename = f"cluster_{cluster_label}__ids.csv"
customer_ids.to_csv(filename, index=False)
```

```
print(f"Saved customer IDs for cluster {cluster_label} to {filename}")
```

**Table 8:**

```
cluster_label = 1

# Filter the DataFrame to get the data for the specified cluster label
cluster_data = new_df[new_df['cluster'] == cluster_label]

# Assuming 'cust_id' is the column name for customer IDs
customer_ids = cluster_data['cust_id']

# Print customer IDs
print(customer_ids)

# Save to CSV file
filename = f"cluster_{cluster_label}__ids.csv"
customer_ids.to_csv(filename, index=False)
print(f"Saved customer IDs for cluster {cluster_label} to {filename}")
```

**Table 9:**

```
cluster_label = 2

# Filter the DataFrame to get the data for the specified cluster label
cluster_data = new_df[new_df['cluster'] == cluster_label]

# Assuming 'cust_id' is the column name for customer IDs
customer_ids = cluster_data['cust_id']

# Print customer IDs
print(customer_ids)

# Save to CSV file
filename = f"cluster_{cluster_label}__ids.csv"
customer_ids.to_csv(filename, index=False)
print(f"Saved customer IDs for cluster {cluster_label} to {filename}")
```

**Clustering:**

```
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

features = new_df[['fresh_beef', 'fresh_poultry', 'deli_cheese', 'ready_to_eat',
       'cheese', 'milk', 'berries_cherries', 'root_veg', 'sweets_in_store',
       'yogurt', 'field_veg', 'salty_snacks', 'packaged_salads', 'cooking_veg',
       'self_serve_deli_meat', 'coffee', 'gourmet_foods', 'baking_ingredients',
       'dairy_natural_foods', 'citrus', 'domestic_beer', 'breads_commercial',
       'ice_cream', 'tomatoes', 'apples', 'service_deli_meat', 'fresh_pork',
       'salad_veg', 'carbonated_soft_drin', 'eggs', 'cereal_rte', 'water',
       'tropical', 'crackers_health_cake', 'fresh_salmon', 'bathroom_tissue',
       'peppers', 'cigarettes', 'butter_margarine', 'sauces_marinades',
       'grapes', 'alternatives_commerc', 'breads_in_store',
       'laundry_household_cl', 'chocolate', 'juices_drinks_refr',
       'domestic_wine', 'confectionary_bars', 'bacon', 'dinner_entres',
       'snacks_natural_foods', 'side_dish_rice', 'frozen_natural_foods',
       'sushi', 'ready_to_heat', 'rolls_in_store', 'cut_fruit', 'bananas',
       'cookies', 'cut_flowers', 'juices_drinks', 'spreads', 'cut_veg',
       'breakfast_natural_fo', 'canned_soup', 'shrimp', 'baking_spices',
       'food_wrap_bags', 'pizza', 'nuts_seeds_snacks', 'baked_protein',
       'fruit', 'condiments', 'nutritional_portable', 'oils',
       'canned_fish_meat', 'live_plant_material', 'beverages_natural_fo',
       'vegetables_frozen', 'sausage', 'baking_bulk_natura',
       'dressing_dips_juices', 'pasta', 'canned_vegetables',
       'meal_makers_natural', 'towels_paper_reu', 'salad_dressings',
       'milk_natural_foods', 'disposable_diapers', 'alternative_beverage',
       'mexican', 'dog_food']]

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)


# # of clusters using the Elbow Method
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(12, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

# k-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(scaled_features)

# cluster labels
new_df['cluster'] = kmeans.labels_

cluster_analysis = new_df.groupby('cluster').mean()
print(cluster_analysis)
```

**Association Rule:**

- **All data:**

```
# convert data to a binary dataframe for associations rule processing
threshold = 0
df = data.drop('cust_id', axis=1).applymap(lambda x: 1 if x > threshold else 0)

# finding frequent itemsets
transactions = df.values.tolist()
frequent_itemsets = apriori(transactions, min_support=0.8, use_colnames=True)

print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
print(rules)
```

- **Cluster 0:**

```
cluster0_ids = pd.read_csv("cluster_0_ids.csv")
cluster0_data = pd.merge(cluster0_ids, data, on='cust_id', how='left')
cluster0_data.head()

# convert data to a binary dataframe for associations rule processing
threshold = 20
df = cluster0_data.drop('cust_id', axis=1).applymap(lambda x: 1 if x > threshold else 0)

# finding frequent itemsets
```

```
frequent_itemsets = apriori(df, min_support=0.85, use_colnames=True)

print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.9)
filtered_rules = rules[rules['consequents'].apply(lambda x: 'domestic_beer' in str(x))] #filter for
rules that has domestic beer in consequent
print(filtered_rules)
```

- **Cluster 1:**

```
cluster1_ids = pd.read_csv("cluster_1__ids.csv")
cluster1_data = pd.merge(cluster1_ids, data, on='cust_id', how='left')
cluster1_data.head()

# convert data to a binary dataframe for associations rule processing
threshold = 0
df = cluster1_data.drop('cust_id', axis=1).applymap(lambda x: 1 if x > threshold else 0)

# finding frequent itemsets
frequent_itemsets = apriori(df, min_support=0.7, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.9)
filtered_rules = rules[rules['consequents'].apply(lambda x: 'domestic_beer' in str(x))] #filter for
rules that has domestic beer in consequent
print(filtered_rules)
```

- **Cluster 2:**

```
cluster2_ids = pd.read_csv("cluster_2_ids.csv")
cluster2_data = pd.merge(cluster2_ids, data, on='cust_id', how='left')
cluster2_data.head()
# convert data to a binary dataframe for associations rule processing
threshold = 20
df = cluster2_data.drop('cust_id', axis=1).applymap(lambda x: 1 if x > threshold else 0)

# finding frequent itemsets
frequent_itemsets = apriori(df, min_support=0.94, use_colnames=True)

print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.95)
```

```
filtered_rules = rules[rules['consequents'].apply(lambda x: 'domestic_beer' in str(x))] #filter for
rules that has domestic beer in consequent
print(filtered_rules)
```

**Logistic Regression:**

- **Cluster 0:**

```
cluster0_ids = pd.read_csv("cluster_0_ids.csv")
cluster0_data = pd.merge(cluster0_ids, non_bud_cust, on='cust_id', how='inner')

buy_mc = pd.read_csv("did_buy_mc_products.csv")
cluster0 = pd.merge(cluster0_data, buy_mc, on='cust_id', how='left')

products_to_consider = {}

for column in cluster0.columns:
    if column.startswith('bought_'):
      column_sum = cluster0[column].sum()

      if column_sum > 0.01 * len(cluster0): # no point recommending products where less than 1% of
customers in the cluster purchase
        products_to_consider[column] = column_sum

for column, column_sum in products_to_consider.items():
    print(f"Column '{column}' has a non-zero sum: {column_sum}")


antecedent_list = [
    'fresh_beef', 'fresh_poultry', 'deli_cheese', 'cheese', 'milk', 'root_veg',
    'field_veg', 'salty_snacks', 'cooking_veg'
]

cluster0_final = cluster0[['cust_id'] + antecedent_list + list(products_to_consider.keys())]

for column in products_to_consider:
    X = cluster0_final[antecedent_list]
    y = cluster0_final[column]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy for {column}: {accuracy}")


# Predictions
cluster0_bud = pd.merge(cluster0_ids, bud_cust, on='cust_id', how='inner')
cluster0_results = pd.DataFrame()

for column in products_to_consider:
    X = cluster0_final[antecedent_list]
    y = cluster0_final[column]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    X_new = scaler.transform(cluster0_bud[antecedent_list])
    y_prob = model.predict_proba(X_new)[:, 1]
    cluster0_results[column] = y_prob

cluster0_results['cust_id'] = cluster0_bud['cust_id']

cluster0_results.to_csv('cluster0_results.csv', index=False)
```

- **Cluster 1:**

```
cluster1_ids = pd.read_csv("cluster_1__ids.csv")
cluster1_data = pd.merge(cluster1_ids, non_bud_cust, on='cust_id', how='inner')

buy_mc = pd.read_csv("did_buy_mc_products.csv")
```

```
cluster1 = pd.merge(cluster1_data, buy_mc, on='cust_id', how='left')

products_to_consider = {}

for column in cluster1.columns:
    if column.startswith('bought_'):
      column_sum = cluster1[column].sum()

      if column_sum > 0.01 * len(cluster1): # no point recommending products where less than 1% of
customers in the cluster purchase
        products_to_consider[column] = column_sum

for column, column_sum in products_to_consider.items():
    print(f"Column '{column}' has a non-zero sum: {column_sum}")


antecedent_list = [
    'fresh_poultry', 'deli_cheese', 'ready_to_eat', 'cheese', 'milk', 'root_veg',
    'field_veg', 'salty_snacks', 'cooking_veg', 'citrus', 'breads_commercial',
    'tomatoes', 'salad_veg', 'eggs', 'bananas'
]

cluster1_final = cluster1[['cust_id'] + antecedent_list + list(products_to_consider.keys())]

for column in products_to_consider:
    X = cluster1_final[antecedent_list]
    y = cluster1_final[column]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy for {column}: {accuracy}")

# Predictions
cluster1_bud = pd.merge(cluster1_ids, bud_cust, on='cust_id', how='inner')
```

```
cluster1_results = pd.DataFrame()

for column in products_to_consider:
   X = cluster1_final[antecedent_list]
   y = cluster1_final[column]

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

   scaler = StandardScaler()
   X_train = scaler.fit_transform(X_train)
   X_test = scaler.transform(X_test)

   model = LogisticRegression(max_iter=1000)
   model.fit(X_train, y_train)

   X_new = scaler.transform(cluster1_bud[antecedent_list])
   y_prob = model.predict_proba(X_new)[:, 1]
   cluster1_results[column] = y_prob

cluster1_results['cust_id'] = cluster1_bud['cust_id']

cluster1_results.to_csv('cluster1_results.csv', index=False)
```

- **Cluster 2:**

```
cluster2_ids = pd.read_csv("cluster_2_ids.csv")
cluster2_data = pd.merge(cluster2_ids, non_bud_cust, on='cust_id', how='inner')

buy_mc = pd.read_csv("did_buy_mc_products.csv")
cluster2 = pd.merge(cluster2_data, buy_mc, on='cust_id', how='left')

products_to_consider = {}

for column in cluster2.columns:
   if column.startswith('bought_'):
     column_sum = cluster2[column].sum()

     if column_sum > 0.01 * len(cluster2): # no point recommending products where less than 1% of
customers in the cluster purchase
       products_to_consider[column] = column_sum

for column, column_sum in products_to_consider.items():
   print(f"Column '{column}' has a non-zero sum: {column_sum}")


antecedent_list = [
   'deli_cheese', 'cheese', 'berries_cherries', 'root_veg', 'field_veg', 'cooking_veg',
   'baking_ingredients', 'citrus'
]

cluster2_final = cluster2[['cust_id'] + antecedent_list + list(products_to_consider.keys())]

for column in products_to_consider:
   X = cluster2_final[antecedent_list]
   y = cluster2_final[column]

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

   scaler = StandardScaler()
   X_train = scaler.fit_transform(X_train)
   X_test = scaler.transform(X_test)

   model = LogisticRegression(max_iter=1000)
   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)

   accuracy = accuracy_score(y_test, y_pred)
   print(f"Accuracy for {column}: {accuracy}")

# Predictions
cluster2_bud = pd.merge(cluster2_ids, bud_cust, on='cust_id', how='inner')
cluster2_results = pd.DataFrame()

for column in products_to_consider:
   X = cluster2_final[antecedent_list]
   y = cluster2_final[column]

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

   scaler = StandardScaler()
   X_train = scaler.fit_transform(X_train)
   X_test = scaler.transform(X_test)

   model = LogisticRegression(max_iter=1000)
   model.fit(X_train, y_train)
   X_new = scaler.transform(cluster2_bud[antecedent_list])
   y_prob = model.predict_proba(X_new)[:, 1]
   cluster2_results[column] = y_prob

cluster2_results['cust_id'] = cluster2_bud['cust_id']

cluster2_results.to_csv('cluster2_results.csv', index=False)
```

**Final Prediction**

```
import pandas as pd
import numpy as np

cluster0_results = pd.read_csv("cluster0_results.csv")
cluster1_results = pd.read_csv("cluster1_results.csv")
cluster2_results = pd.read_csv("cluster2_results.csv")
acse_cust = pd.read_csv("acse_beer_customers.csv")
target_cust = pd.read_csv("acse_beer_customers.csv")


# append the three clusters' results together
all_results = pd.concat([cluster0_results, cluster1_results, cluster2_results], ignore_index=True)

all_results = all_results[~all_results['cust_id'].isin(acse_cust['cust_id'])] #filter out acse customers
all_results = all_results[all_results['cust_id'].isin(target_cust)] #filter out loyal budweiser customers

# reorder columns - have cust_id be the first column
cols = list(all_results.columns)
cols.insert(0, cols.pop(cols.index('cust_id')))
all_results = all_results.reindex(columns=cols)

# fill NAs with 0
all_results.fillna(0, inplace=True)

final_output = []

for index, row in all_results.iterrows():
    # find the column index with the highest probability
    max_prob_col_index = np.argmax(row.values[1:])

    # get the corresponding column name and probability
    max_prob_col_name = all_results.columns[max_prob_col_index + 1]  # +1 to skip the 'cust_id'
column
    max_prob = row[max_prob_col_name]

    final_output.append([row['cust_id'], max_prob_col_name, max_prob])

final_output = pd.DataFrame(final_output, columns=['cust_id', 'product_id', 'predicted_probability'])
final_output['product_id'] = final_output['product_id'].str.replace('bought_', '') # remove 'bought_'
from the product column

# add product description column to output
product_info = pd.read_csv("final_relevant_products_list.csv")
final_output['product_id'] = final_output['product_id'].astype('int64')
final_output = pd.merge(final_output, product_info[['prod_id', 'prod_desc']], left_on='product_id',
right_on='prod_id', how='left')

# reorder columns
final_output.drop(columns=['prod_id'], inplace=True)
final_output = final_output[['cust_id', 'product_id', 'prod_desc', 'predicted_probability']]

final_output.to_csv('final_output.csv', index=False)
```