SAS PROGRAMMING AND MACHINE LEARNING MILESTONE PROJECT

SAS & AUEB's MSc. in Business Analytics
Academic Specialization Joint Certificate

Abstract

This Milestone Project is part of the required procedure for obtaining the SAS Academic Specialization in SAS Programming and Machine Learning. The objective of the project is to apply techniques for accessing, processing, managing, and mining of real-world data and to provide solutions to business problems that today's organizations face using Base SAS Programming and SAS Visual Data Mining and Machine Learning on SAS Viya.

Lefteris Souflas
ele.souflas@aueb.gr

Contents

Table of Figures

Introduction

In today's dynamic business landscape, harnessing the power of data is crucial for informed decision-making and strategic planning. This comprehensive report delves into various facets of an organization's operations, presenting a meticulous analysis of data pre-processing, customer profiling, sales exploration, promotional activities, supplier analysis, RFM customer segmentation, and market basket analysis. Each section unfolds valuable insights that can significantly impact the organization's strategies and enhance overall performance.

1. Data Pre-Processing

Firstly, for every invoice, we calculated the number of Stock Keeping Units (SKU) associated with it. This information is saved in a new dataset called "Invoice Total Items". The first 10 observations from this dataset are printed for a quick overview, as seen in Figure 1.

Obs	Invoice_ID	Invoice_Total_Items
1	1	7
2	2	2
3	3	12
4	4	4
5	5	1
6	6	19
7	7	1
8	8	2
9	9	16
10	10	1

Figure 1 - Total Items per Invoice for the first 10 observations

Then, we determined the total value of products within each invoice. To account for price discounts, we utilized data from the "Promotions" dataset. The result is stored in a new dataset called "Invoice Total Value". Both Sales and Returns invoices were considered for this calculation.

Then, we divided the "Invoice" dataset into two separate tables: "Sales" and "Returns". This division was based on the 'Operation' variable, which specifies whether an invoice is a sale or return. This helps in analyzing sales and returns separately.

Finally, we calculated the ages of customers based on the assumption that today's date is January 1, 2019. Only valid birth years (between 1910 and 2001) were considered. The resulting age values were stored in a new variable, without decimals, making it easier to understand and work with.

These data preprocessing steps provide valuable insights and information for business decision-making. For example, knowing the number of items in each invoice and their total value can help in optimizing inventory and pricing strategies. Separating sales and returns data allows for more focused analysis of customer behavior and satisfaction, and calculating customer ages can support targeted marketing efforts and customer segmentation.

2. Customer Profiling

Then, we began by exploring the demographic characteristics of the customers. This included their age, gender, and region. We used this information to gain a clear understanding of who our customers are and where they came from. The results are as follows:

- Age: We classified our customers into age groups to better understand their distribution. The age groups include:

- <18 -- > “Under 18”
- 18 - 25 -- > “Very Young”
- 26 - 35 -- > “Young”
- 36 - 50 -- > “Middle Age”
- 51 - 65 -- > “Mature”
- 66 – 75 -- > “Old”
- >= 76 -- > “Very Old”

To visualize the distribution of customers across age groups, we created a pie chart. The largest portion of our customers falls into the "Middle Age" group, indicating that this group represents a sizable portion of our customer base, as shown in Figure 2.

- Gender: We assessed the gender distribution of our customers to understand the balance between male and female customers and the results are as shown in Figure 3. We observe that most of our customers are male with a 70-30 percent dominance.

- Region: We analyzed the geographical regions our customers are from to determine if there are regional preferences or variations, as shown in Figure 4. We observe that São Paulo, one of the largest cities in the world, appears to be a region with a significantly high frequency, indicating a substantial number of customers originating from it.

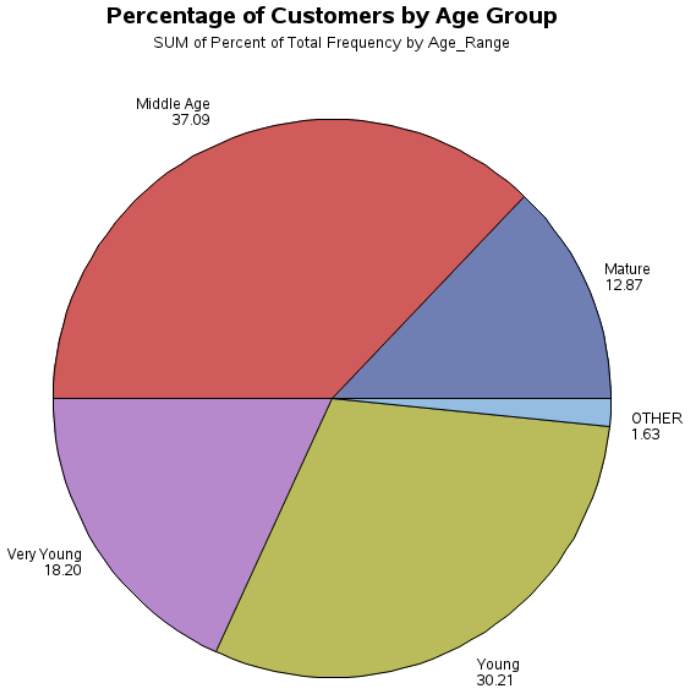

Figure 2 - Percentage of Customers by Age Group

Gender	Frequency	Percent
F	2589	29.93
M	6015	70.07

Figure 3 - Gender Frequency Table

Then, to dive deeper into our customer profiles, we looked at their behavioral characteristics, specifically visits to our stores, the number of distinct products they purchased (SKU count), and the total cost of their purchases. The results are as follows:

- Visits to the Stores: We wanted to understand how often customers visit our stores. Our analysis showed that the "Middle Age" group had the highest number of store visits, as shown in Figure 5.

- Number of Distinct SKUs Purchased: This indicates the diversity of products that our customers are interested in, as shown in Figure 6. We found that all age groups purchase the same variety of products.

- Total Cost of Purchases: We assessed the overall spending of customers in each age group. The "Middle Age" group had the highest total purchase cost, as shown in Figure 7.

Understanding customers' behavior and preferences is crucial in making data-driven decisions for the organization, ensuring we meet their needs effectively and efficiently. These findings help us understand the demographics of the customer base and tailor our products and marketing strategies accordingly. For instance, we might focus on optimizing the shopping experience for the "Middle Age" group, which constitutes a substantial part of the customer base.

Region	Frequency	Percent
AC	10	0.12
AL	62	0.72
AM	24	0.28
AP	7	0.08
BA	357	4.16
CE	197	2.29
DF	260	3.03
ES	161	1.88
GO	250	2.91
MA	88	1.03
MG	805	9.38
MS	89	1.04
MT	100	1.16
PA	98	1.14
PB	78	0.91
PE	214	2.49
PI	65	0.76
PR	500	5.82
RJ	819	9.54
RN	70	0.82
RO	38	0.44
RR	7	0.08
RS	477	5.56
SC	298	3.47
SE	45	0.52
SP	3421	39.85
TO	44	0.51

Figure 4 - Region Frequency Table

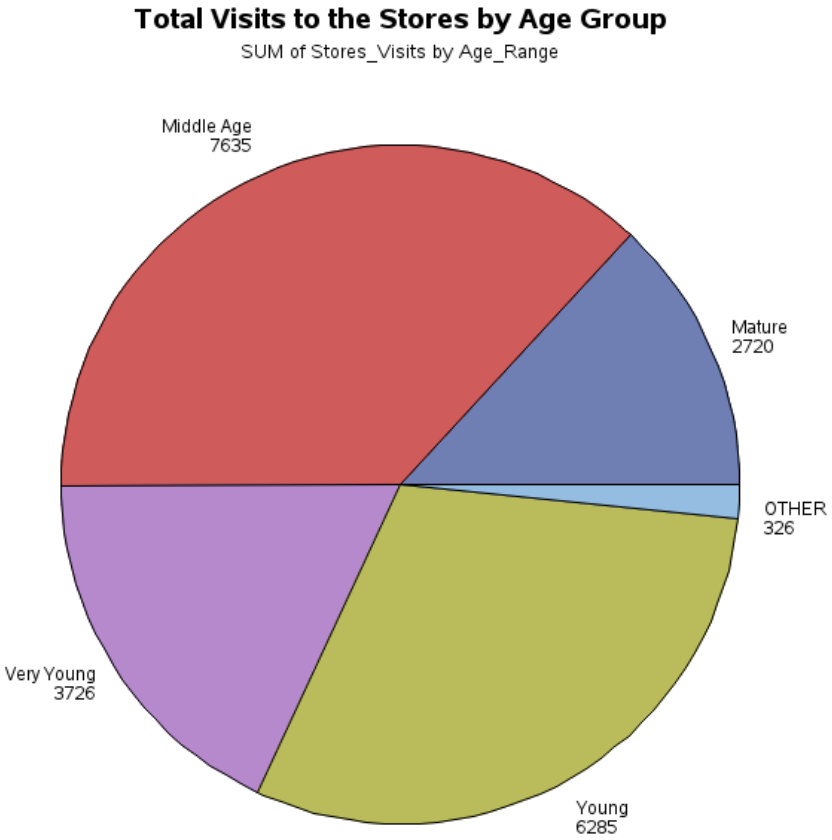

Figure 5 - Total Visits to the Stores by Age Group

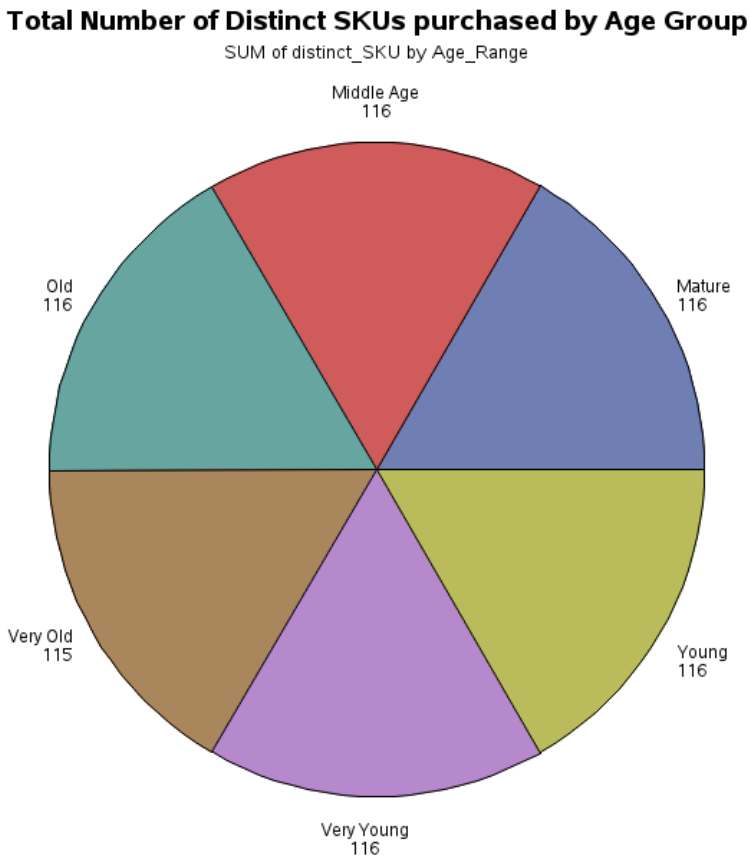

Figure 6 - Total Number of Distinct SKUs purchased by Age Group

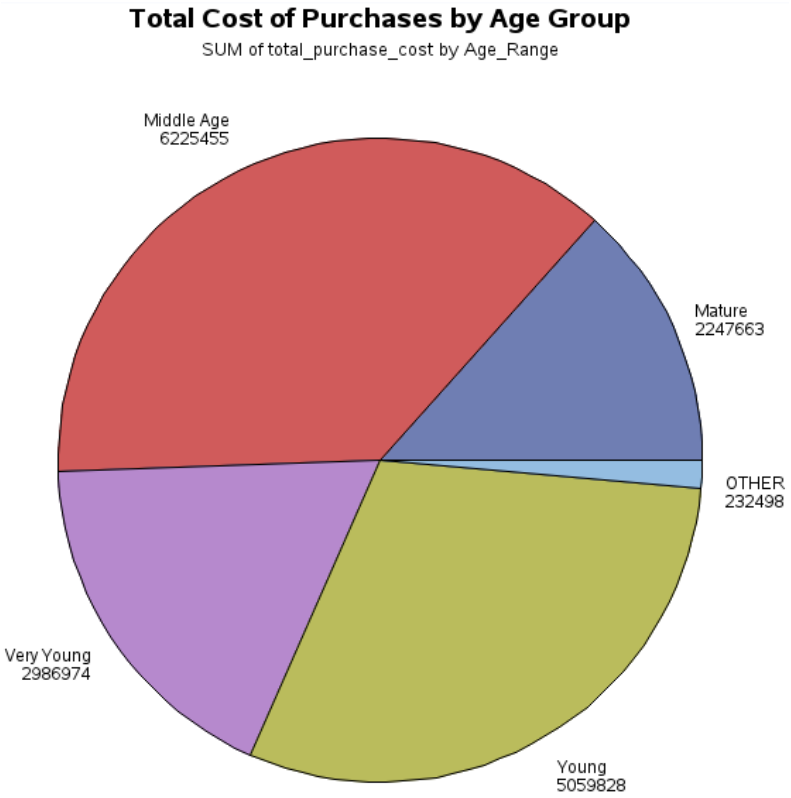

Figure 7 - Total Cost of Purchases by Age Group

3. Exploration and Understanding of Sales

Then, we wanted to explore and understand the organization's sales. Firstly, we compared the level of Sales with the respective of the Returns. It is always crucial to keep track of returns to ensure the company's profitability. From the following bar chart (Figure 8), it is indicated that the organization made significant sales, but there were also returns, which is a common aspect of retail business.

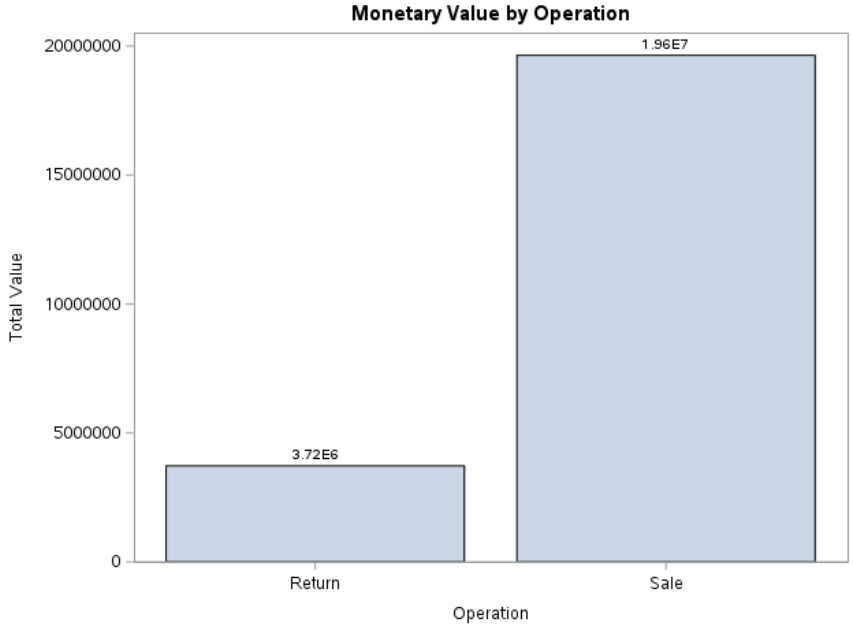

Figure 8 - Bar Chart with the monetary values by operation

Then, we studied the average basket size. From our analysis the following findings were discovered:

- Over time, the average number of SKUs included in a basket (distinct products, disregarding the quantity in each invoice) remained relatively consistent (17), ranging from 16 to 18 SKUs, as shown in Figure 9.
- The average basket monetary value also remained relatively stable over time with slight peaks and valleys, ranging from around 911 to 1,016, averaging on total on 959.17, as shown in Figure 10.
- The payment method chosen by the customers did not affect the average basket size or the average basket monetary value, as shown in Figure 11.

This forementioned stability suggests that customer shopping behaviors did not change significantly during the observed period.

Then, we created a report that presents the top products per product line and product type based on sales value. This report (Figure 12), also provides information on the subtotal sales of each product type, which can help identify the most popular product lines and types and which of them contribute the most to the organization's revenues.

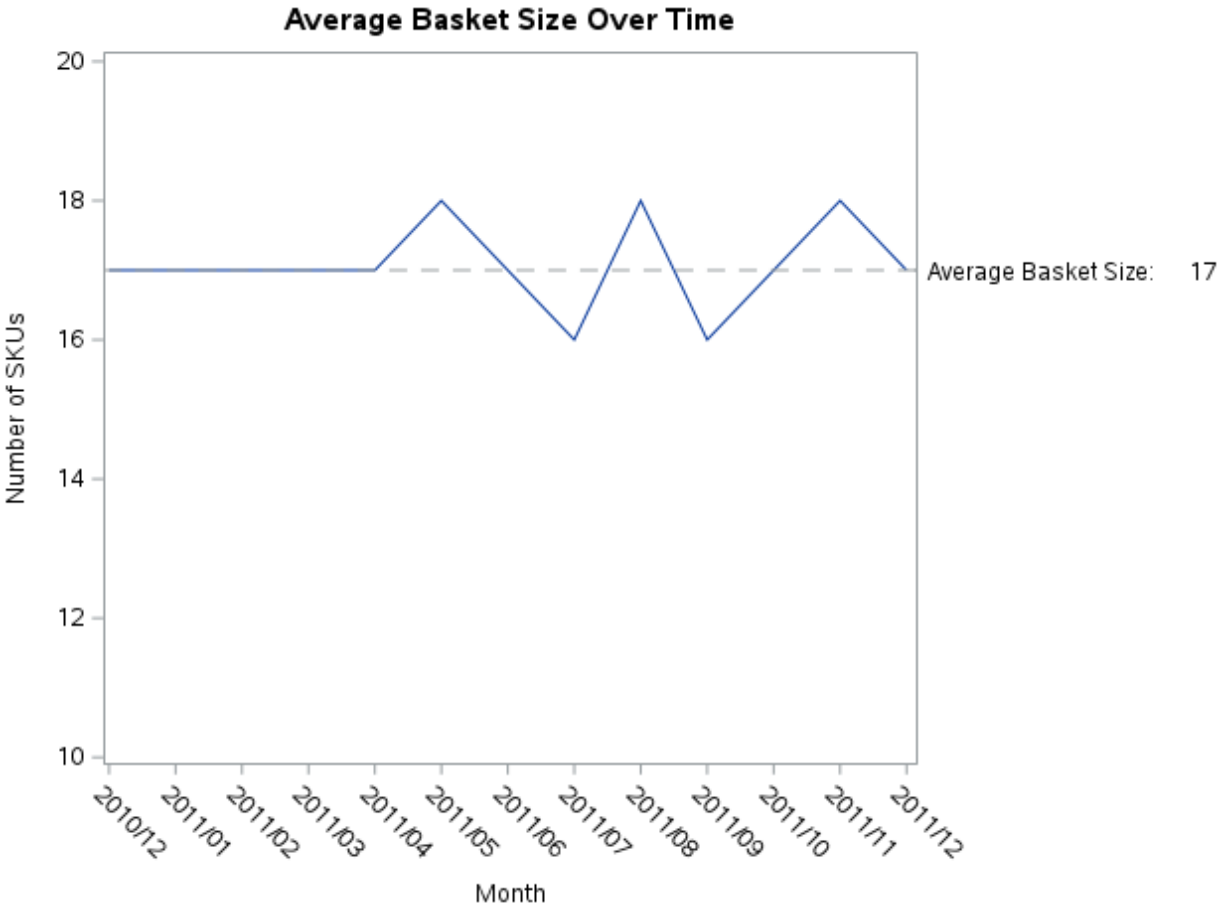

Figure 9 - Average Basket Size Over Time

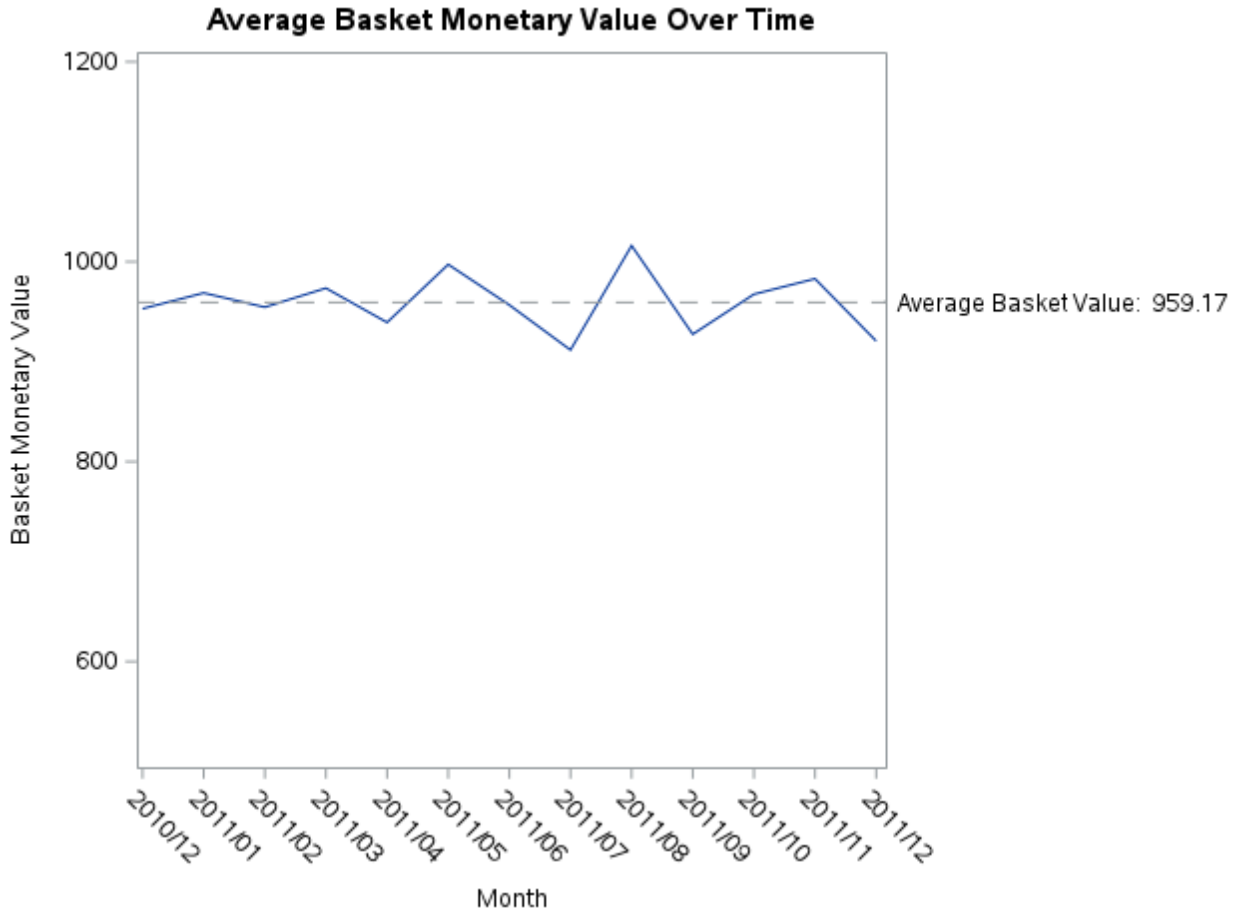

Figure 10 - Average Basket Value over time

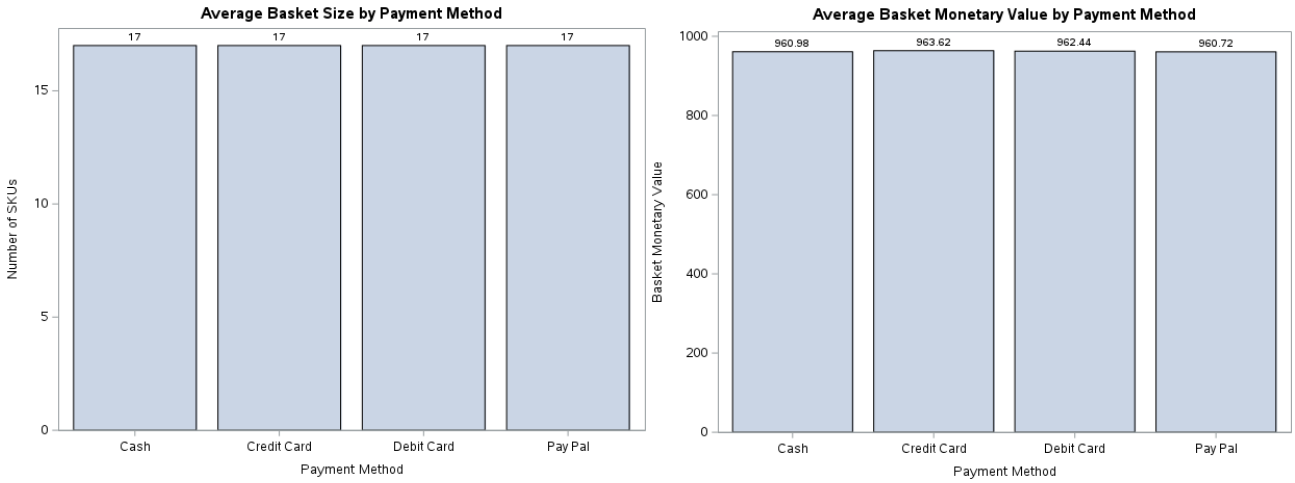

Figure 11 - Average Basket Size and Value by choice of Payment Method

Then, we wanted to understand the contribution to the company's revenues of each region of the country. As shown in the pie chart below (Figure 13), São Paulo (SP) has the highest revenue, approximately 6,635,221. This analysis helps identify the regions with the most significant contribution to revenue and it can inform the organization's marketing and distribution strategies. In Figure 14, we can observe the revenues per region over the entire period under study.

Top Products per Product Line and Product Type

Product line	Product type	Subtotal_Sales	Top_Product	Top_Product_ID	Top_Product_SKU	Top_Product_Sales
Camping Equipment	Cooking Gear	48929	TrailChef Single Flame	7	4055095979496030208	5085
Camping Equipment	Lanterns	58587	Flicker Lantern	41	4055095225197049856	5018
Camping Equipment	Packs	29677	Canyon Mule Climber Backpack	24	4055095531634030080	5185
Camping Equipment	Sleeping Bags	33618	Hibernator Self - Inflating Mat	20	4054202166218109952	4987
Camping Equipment	Tents	28884	Star Gazer 2	13	4052805693123080192	5064
Camping Equipment	~~~SUM~~~	199695	~~~	~~~	~~~	.
Golf Equipment	Golf Accessories	19798	Course Pro Golf and Tee Set	120	9005977191278050304	5012
Golf Equipment	Irons	19562	Hailstorm Steel Irons	109	4027535262979049984	5026
Golf Equipment	Putters	14709	Course Pro Putter	117	8718622611427050496	4993
Golf Equipment	Woods	19809	Hailstorm Steel Woods Set	114	4054202582145050112	5138
Golf Equipment	~~~SUM~~~	73878	~~~	~~~	~~~	.
Mountaineering Equipment	Climbing Accessories	33477	Granite Chalk Bag	138	9005977111278050304	4984
Mountaineering Equipment	Rope	19755	Husky Rope 60	125	9005977181278050304	5012
Mountaineering Equipment	Safety	19740	Husky Harness	129	9005977151278050304	5106
Mountaineering Equipment	Tools	29635	Granite Extreme	144	9005977171278050304	5062
Mountaineering Equipment	~~~SUM~~~	102607	~~~	~~~	~~~	.
Outdoor Protection	First Aid	24525	Calamine Relief	106	4044372023134080000	4952
Outdoor Protection	Insect Repellents	24333	BugShield Natural	94	4055095446550049792	5008
Outdoor Protection	Sunscreen	24012	Sun Shelter 15	101	4053641996585050112	4988
Outdoor Protection	~~~SUM~~~	72870	~~~	~~~	~~~	.
Personal Accessories	Binoculars	29744	Ranger Vision	83	4052906027069069824	5124
Personal Accessories	Eyewear	77976	Polar Wave	58	4027535829976039936	5049
Personal Accessories	Knives	34198	Edge Extreme	73	4053089341760070144	4991
Personal Accessories	Navigation	48777	Trail Master	88	4055095368286020096	5030
Personal Accessories	Watches	63866	Kodiak	53	4027535829815029760	5182
Personal Accessories	~~~SUM~~~	254561	~~~	~~~	~~~	.

Figure 12 - Top products per product line and product type

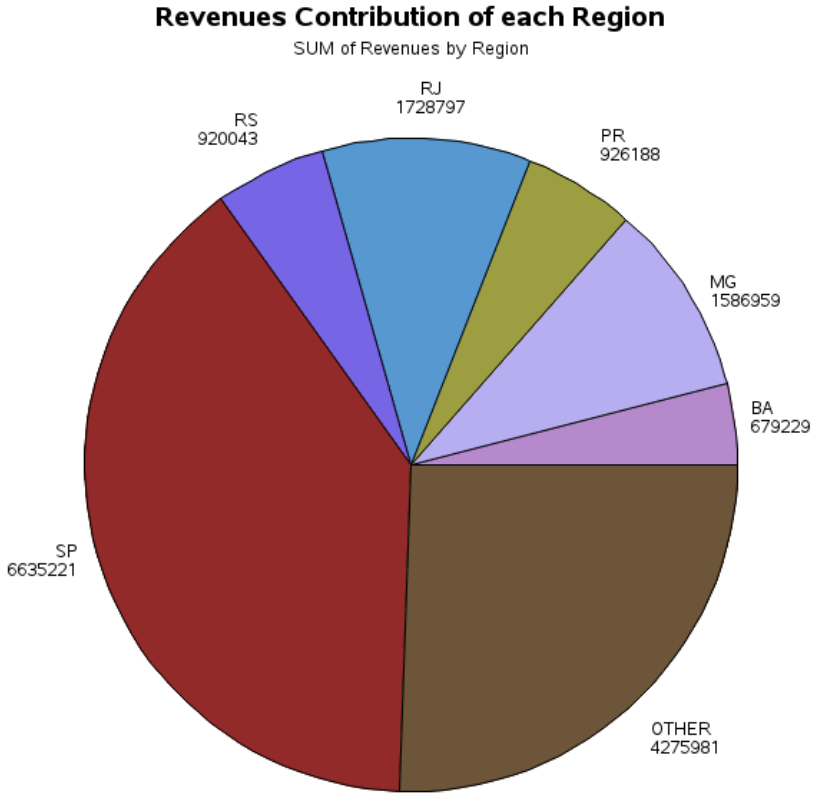

Figure 13 - Revenues contribution of each region

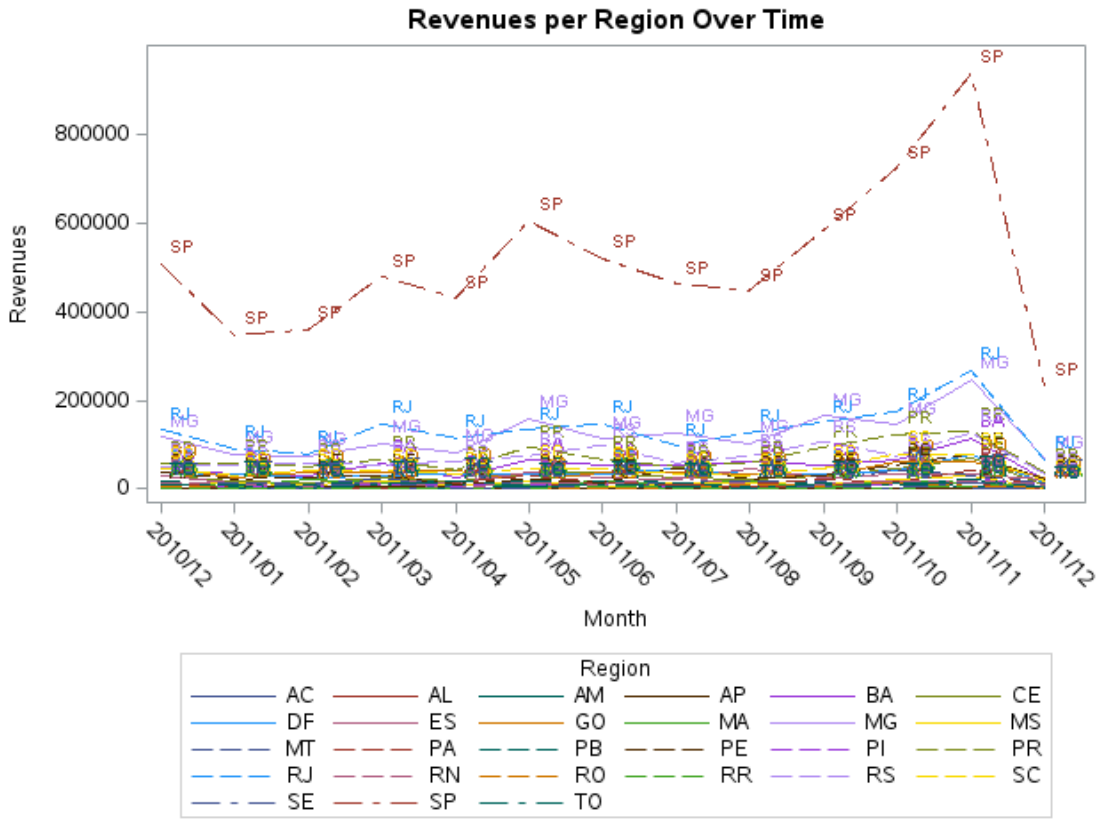

Figure 14 - Revenues per region over time

Finally, for the top region with respect to its contribution to the organization's revenues i.e., São Paulo (SP), we analyzed how much each gender contributes to total revenues, as shown in Figure 15. This information is valuable for tailoring marketing and sales strategies to the specific demographics that contribute the most to the organization's revenues.

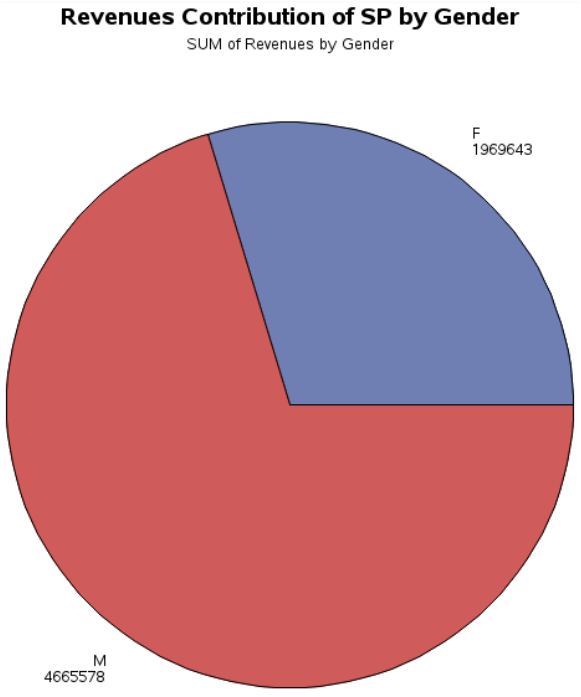

Figure 15 - Gender contribution to São Paulo revenues

In summary, this analysis provides valuable insights into the organization's sales, returns, average basket size, top-performing products, regional revenue contributions, and gender-based revenue contributions. These insights can guide strategic decision-making and help better understand the organization's performance.

4. Promotional Activities Analysis

To understand the impact of promotions on product sales, we first categorized promotions into two groups: "No Promotion" and "Promotion" (which includes 10%, 20%, and 30% discount promotions). We found that approximately 62.5% of products were sold without any promotion, while approximately 37.5% of products were sold with various promotion types, as shown in Figure 16. Among the products sold with promotions i.e., the 37.5% of the whole products population, the products sold with each one of the three promotion types were approximately equally distributed i.e., approximately 12.5% of the whole products population was sold with each of the 10%, 20%, and 30% discount promotions, as shown in Figure 17.

Then, we analyzed the distribution of sales across different days of the week to identify if there are any variations or preferences amongst the customers. To do this, we considered the weekday when the sales took place. From our analysis (Figure 18) we found the following key findings:

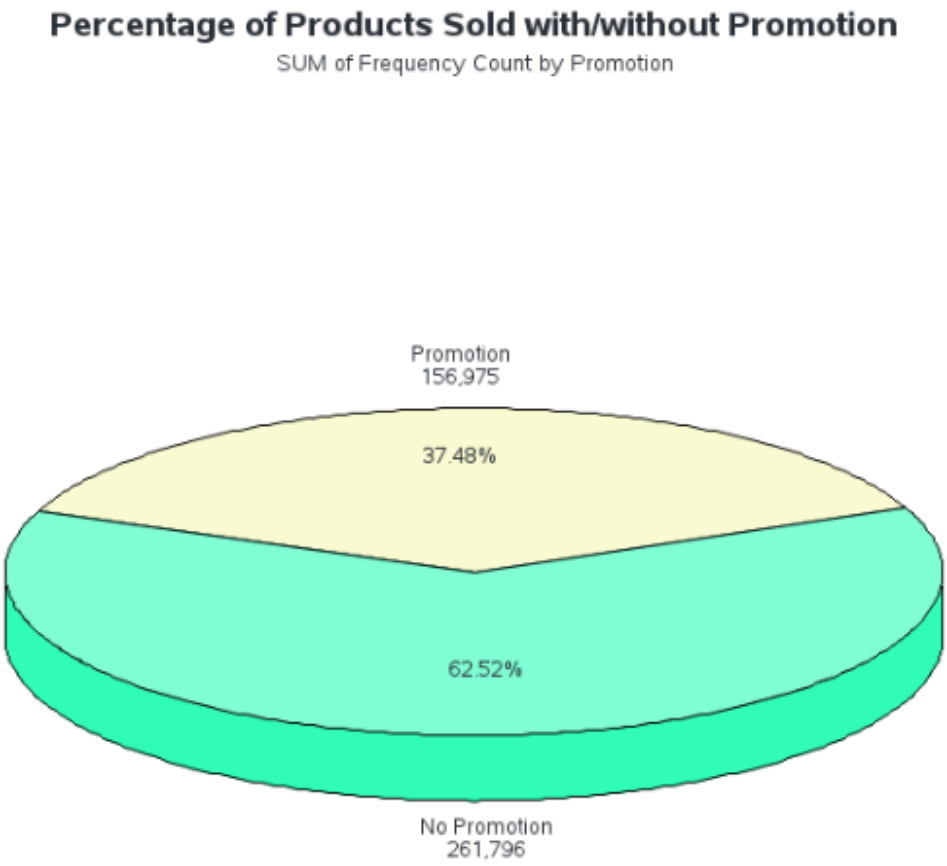

Figure 16 – Three-dimensional pie chart with percentage of products sold with/without promotion

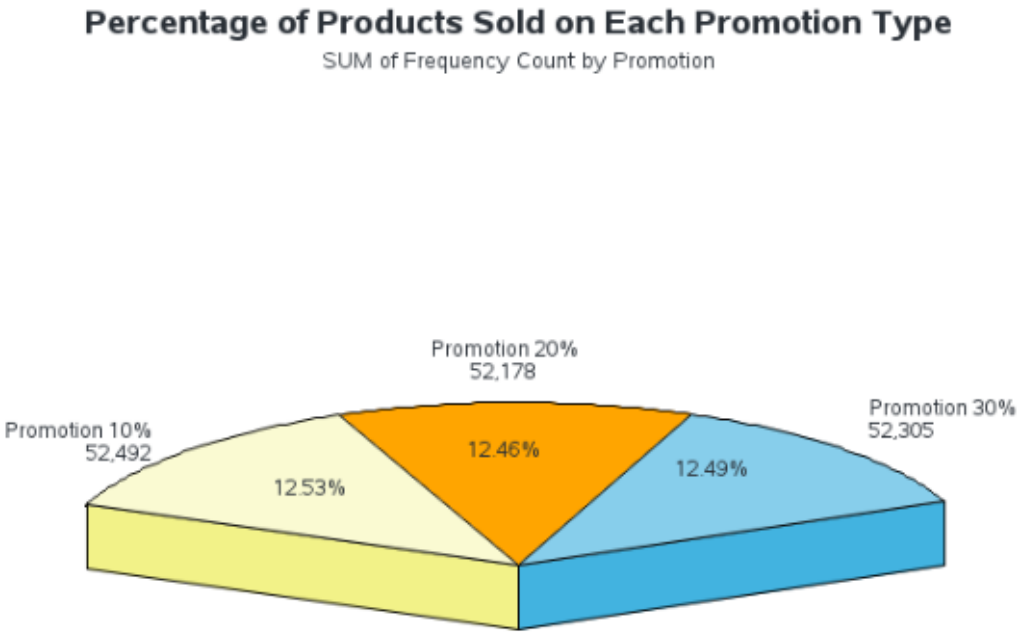

Figure 17 - Three-dimensional pie chart with percentage of products sold on each promotion type

- Saturdays and Sundays accounted for 26.6%, while weekdays (Monday to Thursday) represented 73.4% of the weekly sales transactions, as shown in Figure 19.
- Wednesday was the day with the most weekly transactions (4,348), accounting for 21.3% of the total sales transactions and Saturday was the day with the least weekly transactions (2,215), accounting for 10.9% of the total sales transactions, as shown in Figure 19 and Figure 20.
- The number of distinct SKUs per invoice showed slight variations across different days of the week, as shown in Figure 21.

This information, as well as the overall promotional activities analysis undertaken, can be valuable pieces of the decision-making process, as they highlight the current percentage of products under promotion and the days that are busier for sales, thus, helping the company to plan promotions, allocate resources and guide decisions related to overall business strategies more effectively.

Distribution of Sales per Weekday

Obs	SaleDay	Total_Sale_Transactions	Total_Invoice_Distinct_Items	Average_Invoice_Distinct_Items
1	Saturday	2215	36903	16.6605
2	Sunday	3215	54676	17.0065
3	Monday	3652	62156	17.0197
4	Tuesday	3792	66197	17.4570
5	Wednesday	4348	76413	17.5743
6	Thursday	3197	55738	17.4345

Figure 18 - Distribution of sales per weekday

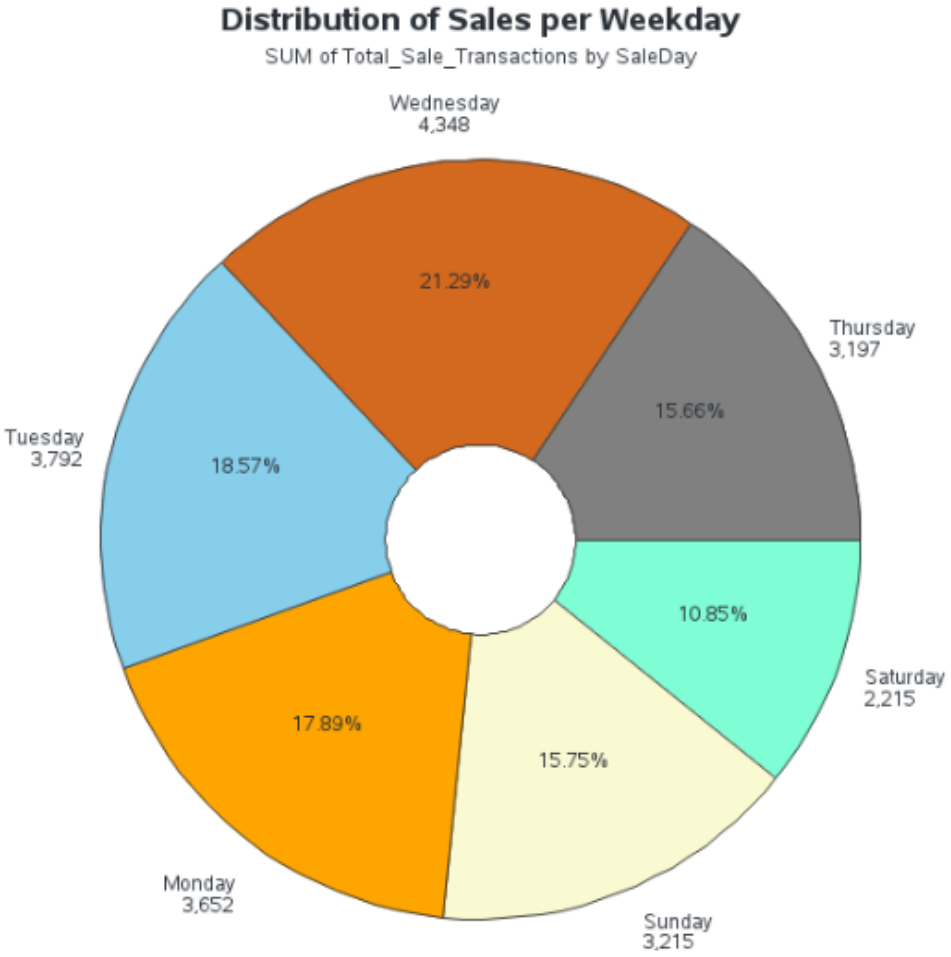

Figure 19 – Donut chart with the distribution of sales per weekday

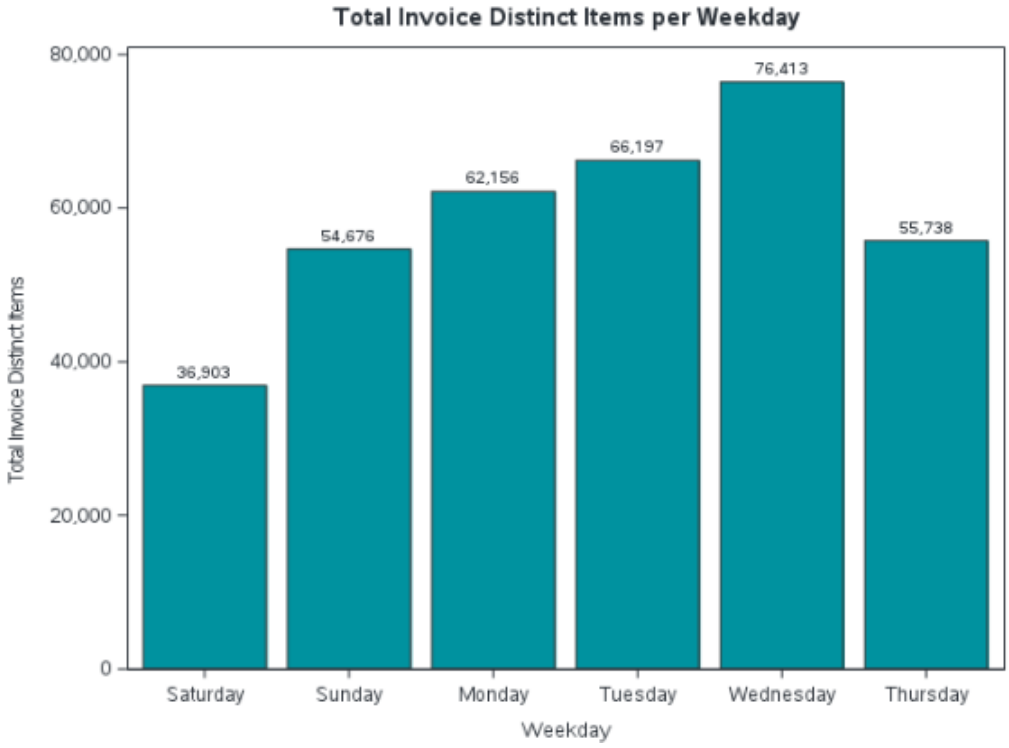

Figure 20 – Bar plot with the total invoice distinct items per weekday

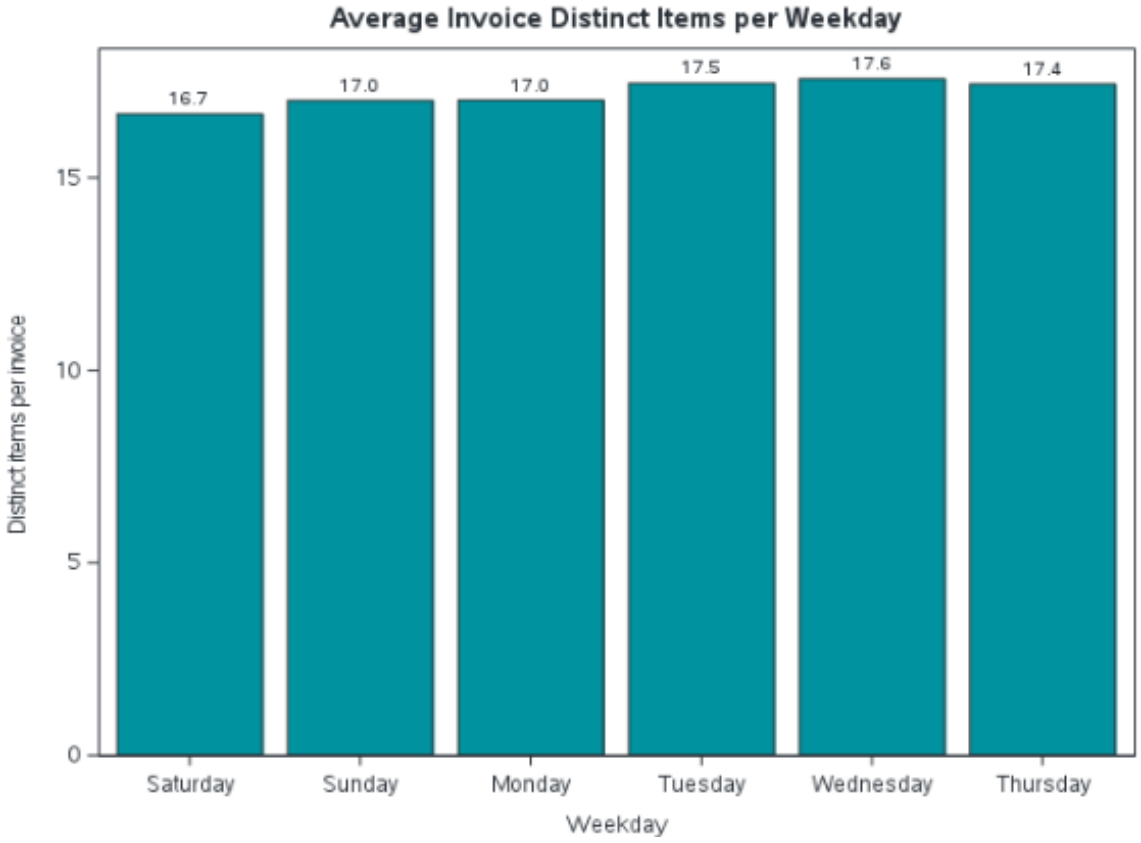

Figure 21 - Bar plot with the average invoice distinct items per weekday

5. Suppliers Analysis

Then, we analyzed the sales data, to gain insights into product demand and identify suppliers with the highest demand. We discovered that the SKU of each product contained valuable information, including the supplier code, which we used to link the product to its supplier. After extracting this information, we conducted a comprehensive analysis of the products sold by each supplier, as shown in the frequency report (Figure 22) and the pie chart (Figure 23) below.

Percentage of Products Sold by Each Supplier

The FREQ Procedure

Supplier_Name		
Supplier_Name	**Frequency**	**Percent**
Carper & Sons	67881	9.65
Dragon SA	112059	15.93
Easy Creator	59063	8.39
Elegance SA	82730	11.76
Fabulo Ltd	73777	10.49
Maestri & Maestri	78348	11.14
Selector Ltd	63538	9.03
Toktai & Chen	101934	14.49
Viking Quality	64281	9.14

Figure 22 - Frequency report of the products sold by each supplier

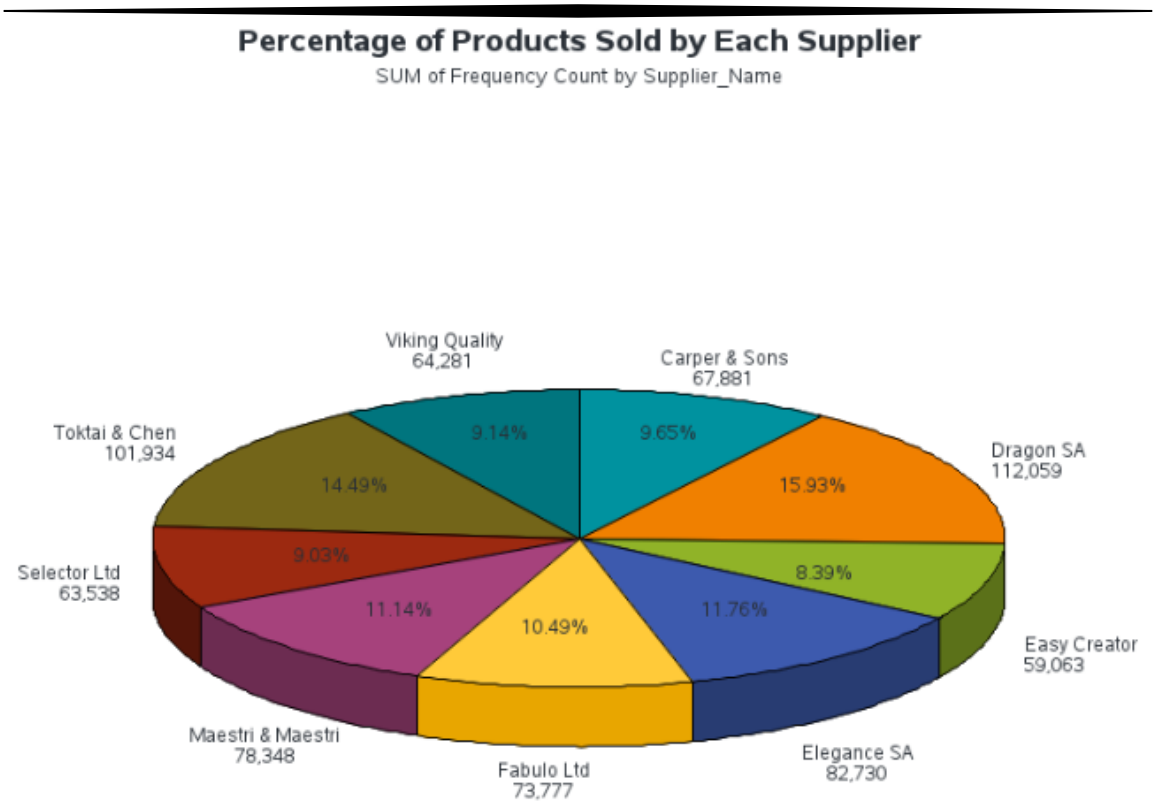

Figure 23 - Three-dimensional pie chart with the percentage of products sold by each supplier

We then analyzed the revenues made by each supplier, as understanding the monetary impact of each supplier is of utmost importance. We examined both the percentage and the actual revenues of products sold by each supplier. From our analysis, as shown in Figure 24 and Figure 25, we found out that:

- Dragon SA emerged as the leading and most lucrative supplier, making a substantial contribution to the overall revenue, indicating robust demand for their products.
- Toktai & Chen, Maestri & Maestri, and Elegance SA collectively represent a considerable proportion of products sold, combining with Dragon SA to account for over 50% of the total revenues generated.

We further analyzed the total revenues of the company concerning the origins of the products sold by each supplier. This information provides a comprehensive view of the company's performance across different product origins and supplier contributions. For displaying our analysis, we provide a cross-tabulation table, which provides a holistic view of revenue distribution based on product origin and supplier. Please refer to the attached cross-tabulation table (Figure 26) for a detailed breakdown of total revenue by product origin and supplier.

In summary, this analysis offers valuable insights into supplier performance, product demand, and revenue distribution. These insights can guide strategic decisions, supplier relationship management, and resource allocation to optimize overall business performance.

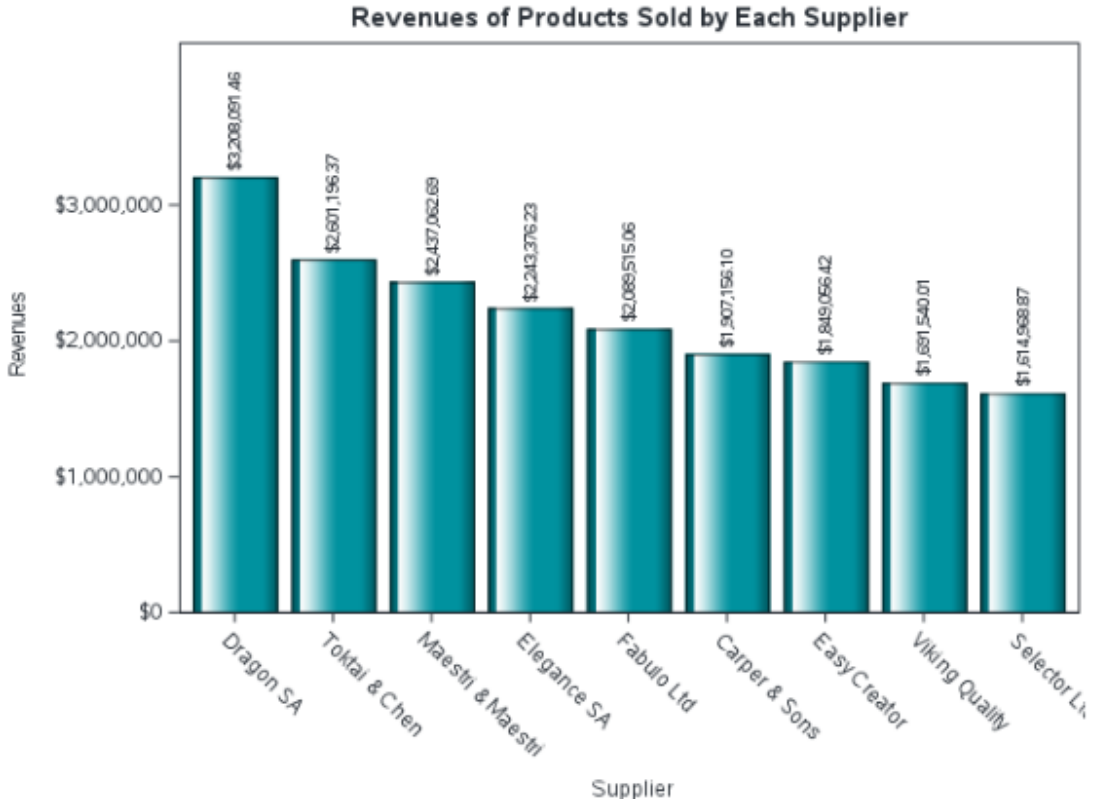

Figure 24 - Bar chart with the revenues of products sold by each supplier

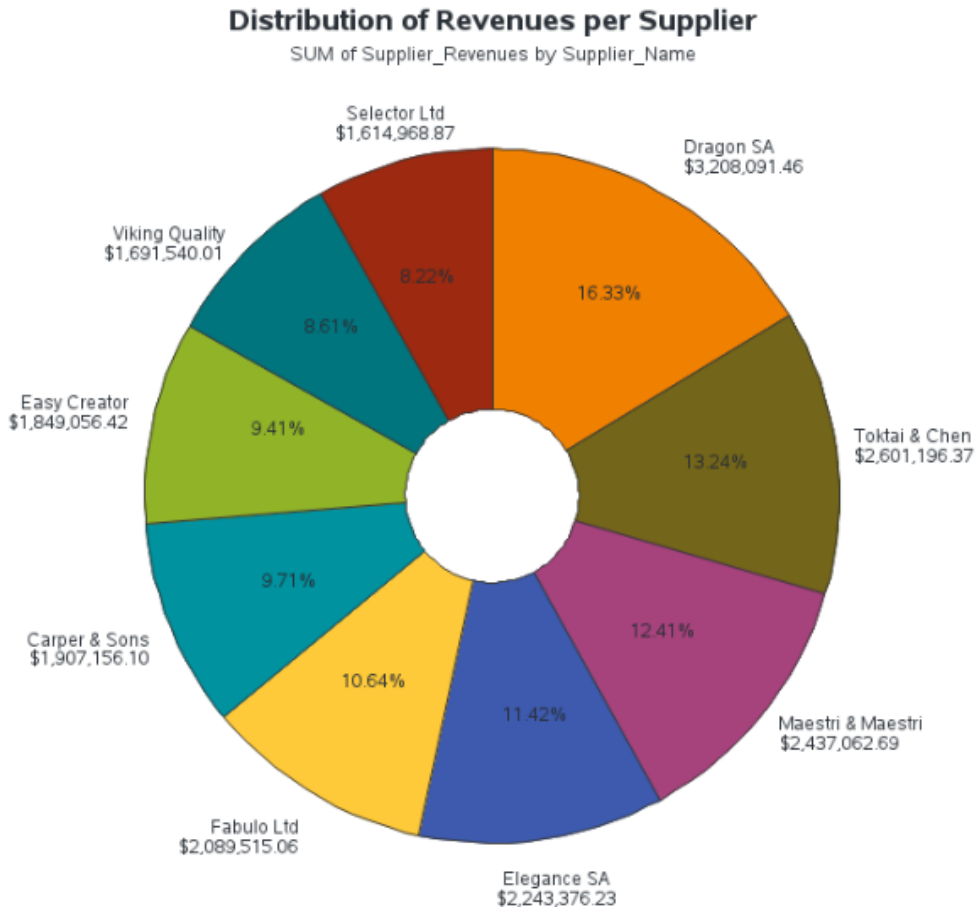

Figure 25 - Donut chart with the distribution of revenues per supplier

Total Revenue by Product Origin & Supplier

Country of Origin	Supplier									Total
	Carper & Sons	Dragon SA	Easy Creator	Elegance SA	Fabulo Ltd	Maestri & Maestri	Selector Ltd	Toktai & Chen	Viking Quality	
China	$508,329.57	$683,497.08	$315,823.87	$864,742.90	$446,780.83	$57,898.68	$77,395.47	$712,200.40	$263,723.69	$3,930,392.48
India	$660,210.73	$589,802.54	$675,353.73	$80,364.94	$780,090.19	$595,112.95	.	$477,316.03	$340,095.44	$4,198,346.54
Spain	$316,859.95	$759,035.79	$295,690.52	$240,771.04	$140,815.44	$878,137.75	$748,051.59	$559,661.46	.	$3,939,023.56
Turkey	$211,729.81	$542,735.58	$111,822.18	$488,021.86	$200,616.21	$526,470.73	$451,479.84	$336,295.42	$527,318.62	$3,396,490.26
US	$210,026.04	$633,020.47	$450,366.12	$569,475.49	$521,212.40	$379,442.58	$338,041.96	$515,723.06	$560,402.26	$4,177,710.37
Total	$1,907,156.10	$3,208,091.47	$1,849,056.42	$2,243,376.23	$2,089,515.06	$2,437,062.69	$1,614,968.87	$2,601,196.37	$1,691,540.01	$19,641,963.21

Figure 26 - Cross tabulation table of total revenue by product origin & supplier

6. Recency-Frequency-Monetary (RFM) Data

In our quest to profile our customers based on their importance, so as to offer them personalized services and products, we have embarked on a comprehensive customer segmentation journey using the RFM model. The RFM model, focusing on Recency, Frequency, and Monetary Value, enables us to discern and cater to the unique needs of each customer. The key metrics of the RFM model are the following:

- Recency (R): Indicates the number of weeks since the customer's last purchase.
- Frequency (F): Reflects how often a customer makes a purchase.
- Monetary Value (M): Represents the total spending by each customer.

Because RFM Customer Segmentation is based on behavioral characteristics i.e., the interaction data of the customers, we kept only customers that had transactional data in the period of our analysis. A sample of 10 customers' RFM data are presented in Figure 27.

This insightful RFM analysis serves as a foundational step towards personalized customer engagement. Understanding the recency, frequency, and monetary patterns empowers us to tailor offerings, thereby enhancing customer satisfaction and loyalty.

Sample of 10 Customers' RFM Data

Customer_ID	R	F	M
1	2	4	1201.65
2	34	1	33.31
3	2	1	982.81
6	5	1	849.03
8	39	1	105.58
9	5	3	2714.71
10	29	2	1187.96
11	5	1	139.16
13	21	3	1046.61
16	38	2	264.38

Figure 27 - Sample of 10 customers' RFM data

7. RFM Customer Segmentation

After the acquisition of RFM data, an intricate process of customer segmentation was undertaken through the utilization of SAS Visual Data Mining and Machine Learning. Employing observation-based clustering, we meticulously categorized customers based on their Recency, Frequency, and Monetary (RFM) data. The outcomes of this segmentation, illustrated in Figure 28, yielded significant insights, summarized as follows:

- The customer base was effectively segmented into five distinct clusters.
- The 1st cluster, comprised of 1,775 customers, spent on average 1,333.61 in a single transaction in one year period (12/2010 – 12/2011) and their last transactions was on average approximately before 9 months (35 weeks), so we can consider them as potential lapsing customers.
- The 2nd cluster consists of 2,083 customers that on average spent 2,544.63 on 3 transactions in a one-year period and their last transaction was on average before 1.5 months (6 weeks).

- The 3rd cluster, comprised of 2,021 customers, spent on average 751.84 in a single transaction in one year period and their last transaction was on average before 2 months (9 weeks).
- The 4th cluster consists of 913 customers that on average spent 2,620.16 on 4 transactions in one year period and their last transaction was on average before 1 month (5 weeks).
- The 5th cluster, comprised of 652 customers, spent on average 7,934.23 on 4 transactions in a one-year period and their last transaction was on average before 1.5 months (5.5 weeks), so we can consider them as our VIP customers (they spent on average more than all other customer segments combined).

Clusters Aggregated Results

Cluster ID	Population	Average_Recency	Average_Frequency	Average_Monetary
1	1775	35	1	1333.61
2	2083	6	3	2544.63
3	2021	9	1	751.84
4	913	5	4	2620.16
5	652	5.5	4	7934.23

Figure 28 - Customers Segmented Profiles

Notably, the two most important clusters created are the potential lapsing and the VIP segments of customers i.e., the 1st and 5th clusters, respectively. Visualization of the clusters is provided through the following charts with the parallel coordinates (Figure 29) and the boxplots of the RFM values (Figure 30, Figure 31, and Figure 32) of the two clusters. We can observe that, within each cluster, there exist customers that differentiate a little from the representative average customer of the cluster. However, 50% of the customers of each cluster, which is denoted by the rectangular colored box in the boxplot, share a great distance, presenting the significant difference of the two customer profiles.

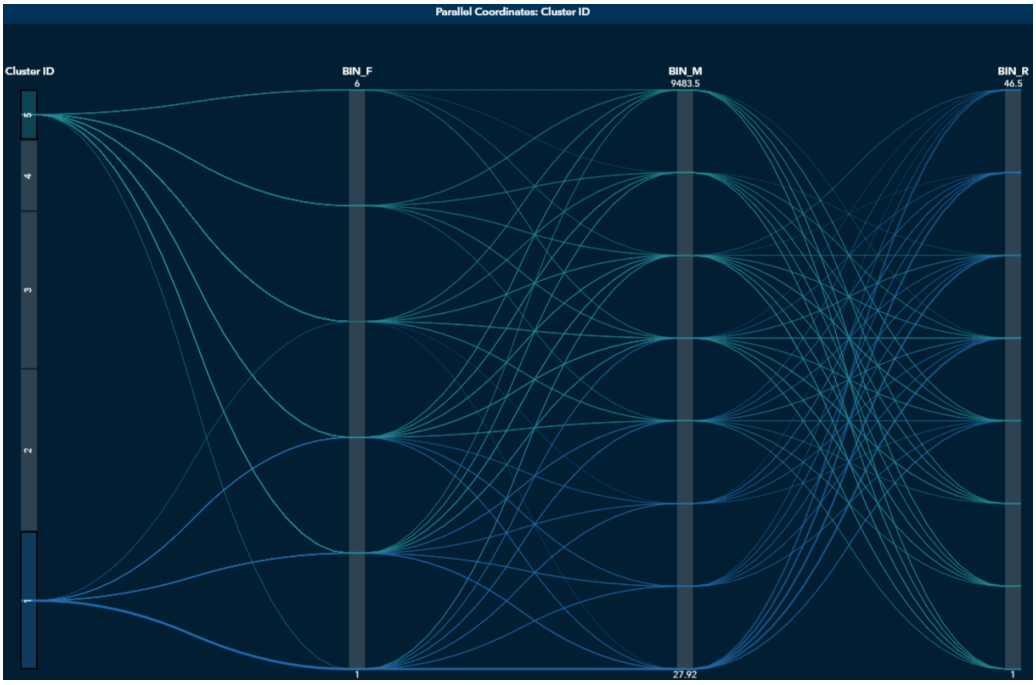

Figure 29 - Parallel Coordinates Graph with most bold lines of higher F and M values and lower R value of the 5th (VIP) cluster

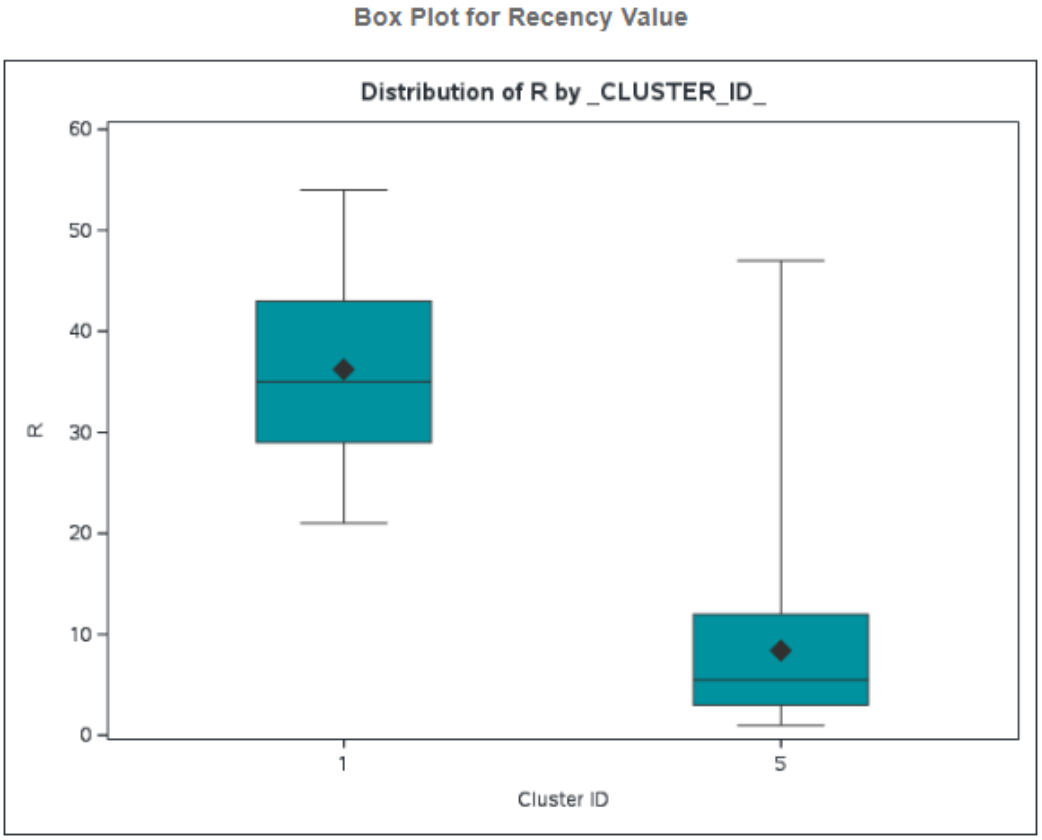

Figure 30 - Recency value boxplot of the two clusters

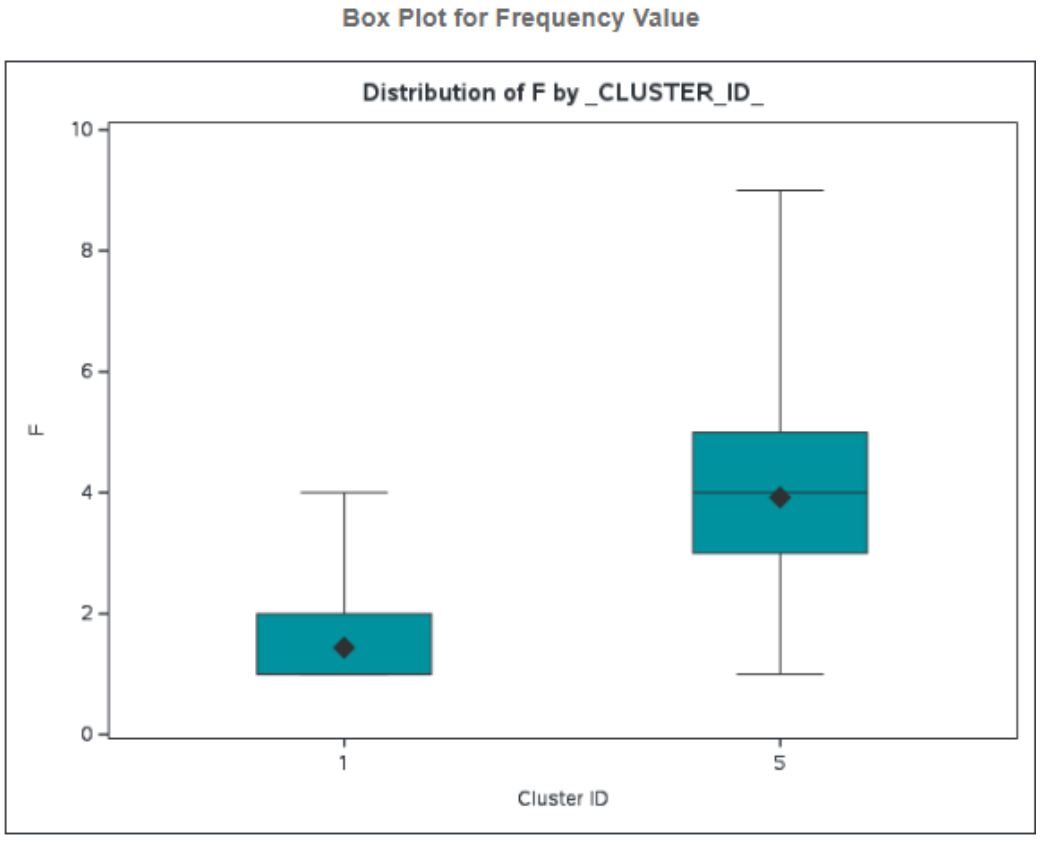

Figure 31 - Frequency value boxplot of the two clusters

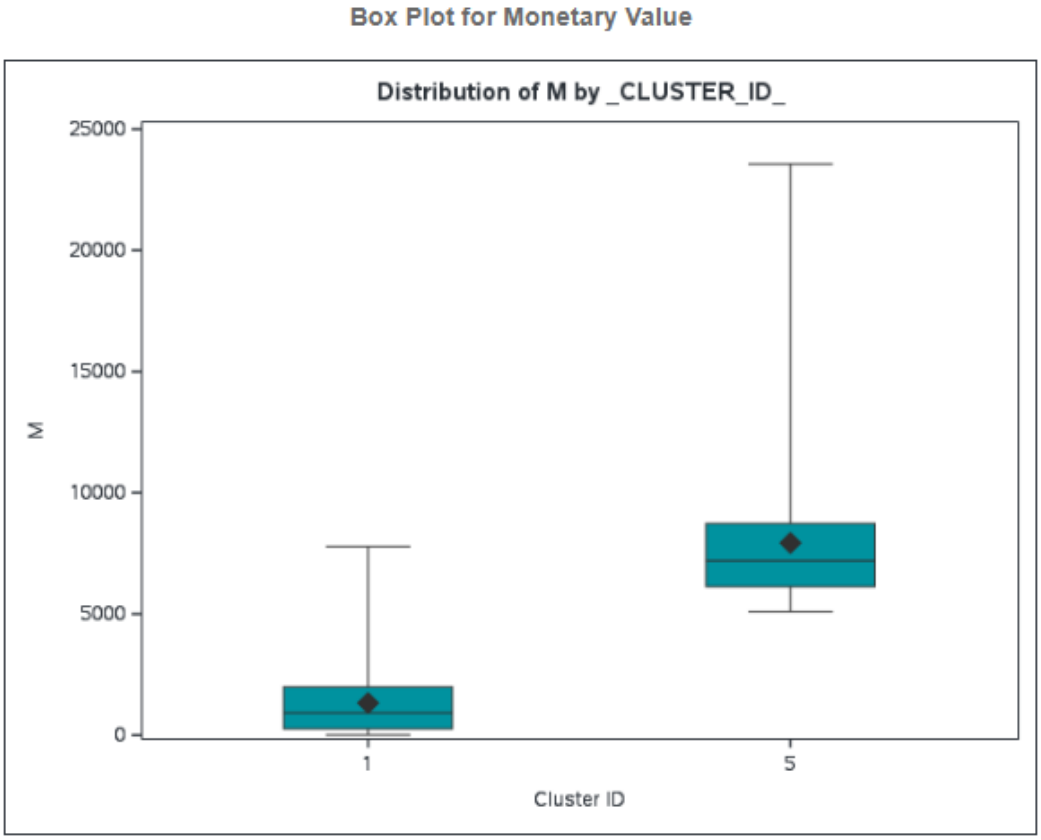

Figure 32 - Monetary value boxplot of the two clusters

While distinct behavioral patterns characterize the two customer profiles, demographic attributes exhibit minimal disparities, as can be observed in the figures below (Figure 33, Figure 34, and Figure 35). The geographic analysis of Figure 35 reveals a singular distinction in the 5th cluster, where customers from Acre (AC), Amapá (AP), and Rondônia (RO) States of Brazil are absent compared to the 1st cluster.

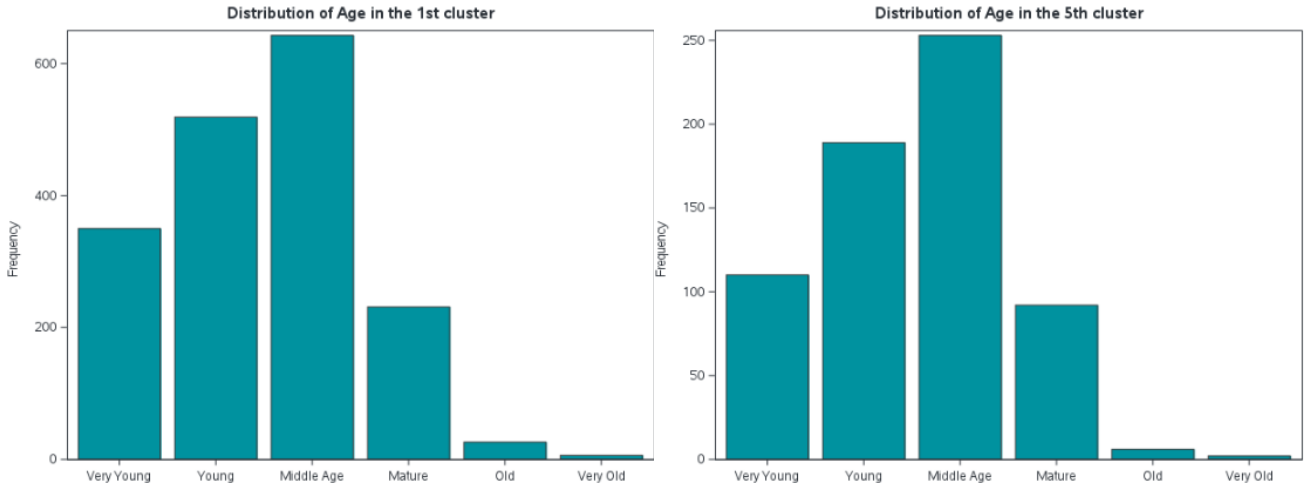

Figure 33 - Distribution of Age in the two clusters

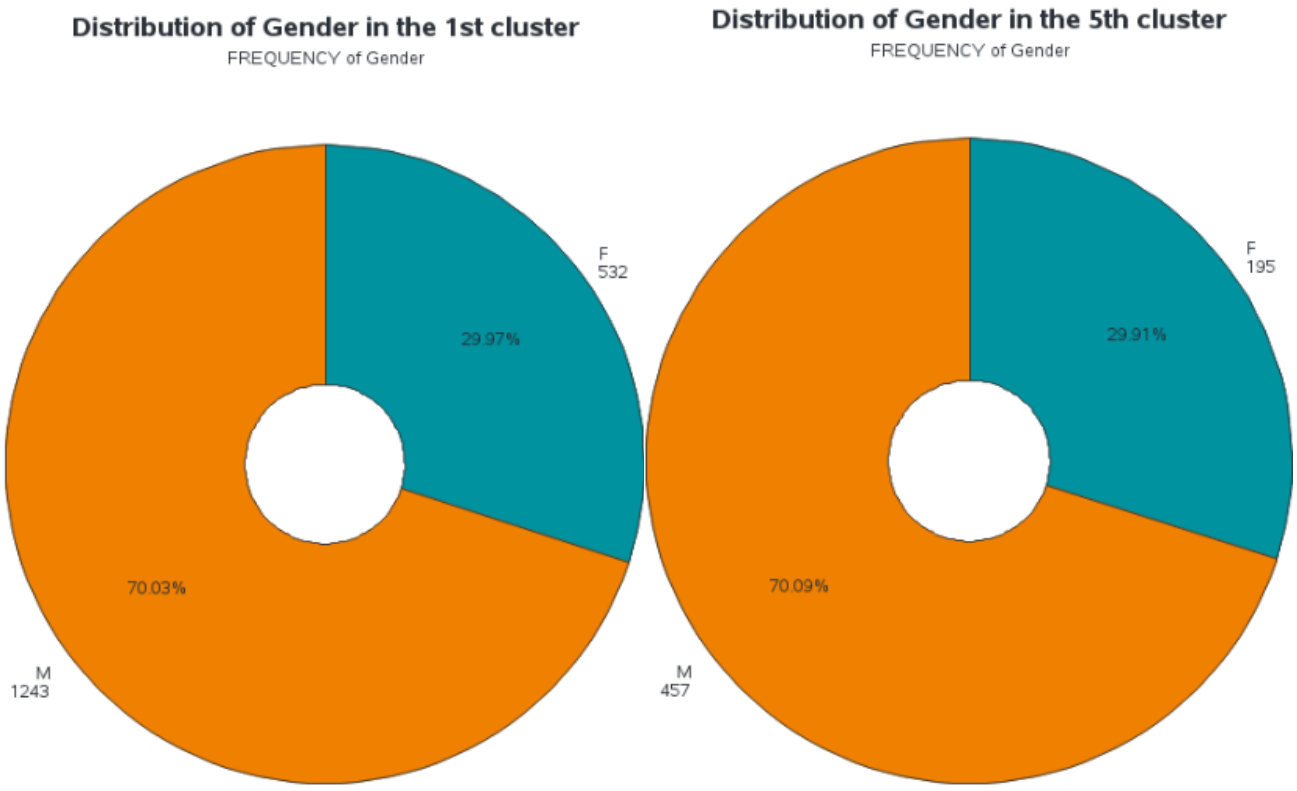

Figure 34 - Distribution of Gender in the two clusters

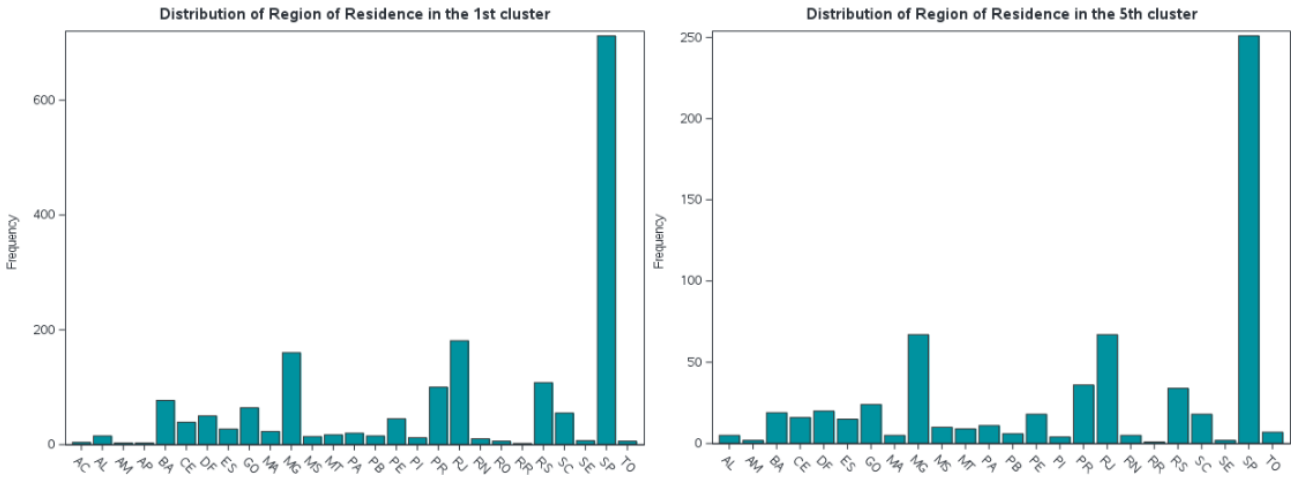

Figure 35 - Distribution of Residence in the two clusters

In conclusion, the meticulous examination of RFM data, coupled with sophisticated clustering techniques, has facilitated a nuanced understanding of customer segments, paving the way for targeted strategies tailored to the unique characteristics of each cluster.

8. Market Basket Analysis

Finally, a Market Basket Analysis was conducted to identify associations among product categories based on sales transactions within the dataset. Additionally, we extended the analysis to focus on the two most important customer clusters previously identified.

The initial analysis encompassed the entire dataset, excluding returns, to ensure insights are derived from positive customer interactions. The results of this analysis, stored in the "MBA Results" dataset, provide a comprehensive overview of product category associations. A sample of the top 10 product category associations based on the lift of each association rule can be observed in Figure 36.

Top 10 Product Categories Associations

RULE	LIFT
Safety & Woods ==> Golf Accessories & Putters	2.93
Golf Accessories & Putters ==> Safety & Woods	2.93
Putters & Rope ==> Irons & Safety	2.92
Irons & Safety ==> Putters & Rope	2.92
Putters & Rope ==> Safety & Woods	2.91
Safety & Woods ==> Putters & Rope	2.91
Safety & Woods ==> Irons & Putters	2.90
Irons & Putters ==> Safety & Woods	2.90
Irons & Woods ==> Golf Accessories & Putters	2.90
Golf Accessories & Putters ==> Irons & Woods	2.90

Figure 36 - Top 10 product category associations in the entire dataset

Then, from the two subsets of data created for customers belonging to the two most important clusters, the associations among product categories within these clusters were analyzed separately to tailor recommendations based on cluster-specific preferences. A sample of the top 10 product category associations based on the lift of each association rule in each cluster can be observed in Figure 37 and Figure 38.

Top 10 Product Categories Associations in the 1st Cluster

RULE	LIFT
Insect Repellents & Sunscreen ==> Golf Accessories & Woods	3.04
Golf Accessories & Woods ==> Insect Repellents & Sunscreen	3.04
Safety & Sunscreen ==> Golf Accessories & Woods	3.01
Golf Accessories & Woods ==> Safety & Sunscreen	3.01
Safety & Sunscreen ==> Putters & Woods	2.96
Putters & Woods ==> Safety & Sunscreen	2.96
Sunscreen & Tents ==> Golf Accessories & Woods	2.94
Golf Accessories & Woods ==> Sunscreen & Tents	2.94
Irons & Sunscreen ==> Golf Accessories & Woods	2.93
Golf Accessories & Woods ==> Irons & Sunscreen	2.93

Figure 37 - Top 10 product category associations in the 1st cluster

Top 10 Product Categories Associations in the 2nd Cluster

RULE	LIFT
Putters & Rope ==> Golf Accessories & Safety	1.87
Golf Accessories & Safety ==> Putters & Rope	1.87
Putters & Rope ==> Safety & Woods	1.86
Safety & Woods ==> Putters & Rope	1.86
Putters & Woods ==> Golf Accessories & Rope	1.86
Golf Accessories & Rope ==> Putters & Woods	1.86
Putters & Rope ==> Insect Repellents & Safety	1.85
Insect Repellents & Safety ==> Putters & Rope	1.85
Putters & Woods ==> Golf Accessories & Safety	1.84
Golf Accessories & Safety ==> Putters & Woods	1.84

Figure 38 - Top 10 product category associations in the 2nd cluster

From the results of the market basket analysis conducted, utilizing the whole dataset insights could optimize product placement and promotions across all stores based on associations identified in the whole dataset analysis. The company could also leverage the insights from cluster-specific analyses to tailor proposals and offers for customers in the two most important clusters identified previously.

To sum up, the Market Basket Analysis provides a valuable foundation for understanding customer behavior and preferences. By incorporating these insights into our marketing strategies, we can enhance customer satisfaction and drive sales in a targeted and efficient manner.

Conclusion

In the ever-evolving landscape of today's organizations, the insights gleaned from this comprehensive analysis serve as a compass, guiding towards strategic decisions that resonate with the pulse of the organization's customers and market dynamics. The journey began with meticulous data pre-processing, laying the foundation for informed decision-making in inventory management, pricing strategies, and customer segmentation.

The exploration of customer profiles unfolded a tapestry of demographics and behaviors, offering a nuanced understanding of the customer base. The scrutiny of sales data provided a panoramic view of the organization's performance, from regional revenue contributions to the impact of promotions on product sales. Supplier analysis shed light on product demand and supplier contributions, offering a roadmap for supplier relationship management and resource allocation. The Customer Segmentation based on the Recency-Frequency-Monetary (RFM) Model opened a window into distinct customer clusters, enabling personalized engagement and enhancing customer satisfaction and loyalty. The culmination of this journey, the Market Basket Analysis, unlocks associations among product categories, laying the groundwork for targeted product placements and promotions. Cluster-specific analyses refine our approach, ensuring that marketing strategies resonate with the most important customer segments.

In essence, this report is not merely a compilation of data; it's a strategic blueprint. Armed with these insights, we can navigate the competitive landscape with precision, making decisions that foster growth, enhance customer satisfaction, and position the organization for sustained success in the marketplace.

Appendix

Data Upload Challenges and SAS Code

Because we faced difficulty in accessing our already uploaded data from previous period in the SAS Studio in SAS Viya for Learners platform, we tried to re-upload the data, however, without success, as it is shown in Figure 39.

As an alternative, we tried SAS Studio in SAS On-Demand for Academics, as it was suggested in the SAS Viya for Learners platform (Figure 40) and it worked successfully, so we continued using the latter platform i.e., SAS ODA, with the same credentials as in SAS Viya, until the former errors (SAS Viya) were fixed (from fourth task and onwards).

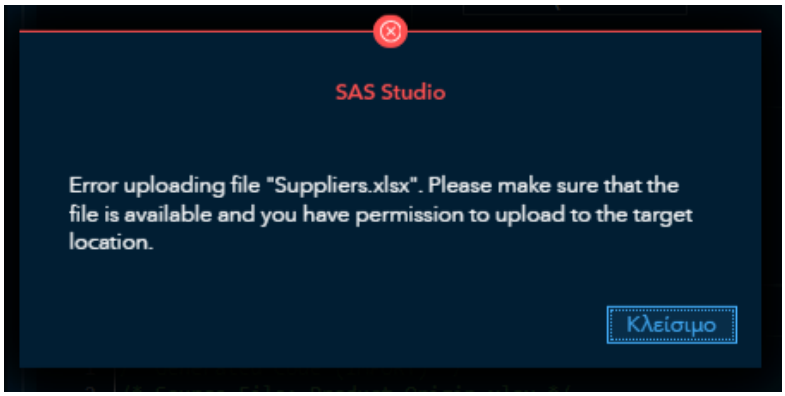

Figure 39 - Error uploading data in SAS Viya for Learners

If you are having difficulty accessing or uploading data in SAS Studio in SAS Viya for Learners, we apologize for the inconvenience. The SAS VFL team is working to resolve the issue as quickly as possible. As an alternative, you may try using SAS Studio (9.4 version; not Viya) in SAS On-Demand for Academics. Please note that some SAS procedures may not be available in SAS ODA.

Figure 40 - Info Message in SAS Viya suggesting using SAS ODA

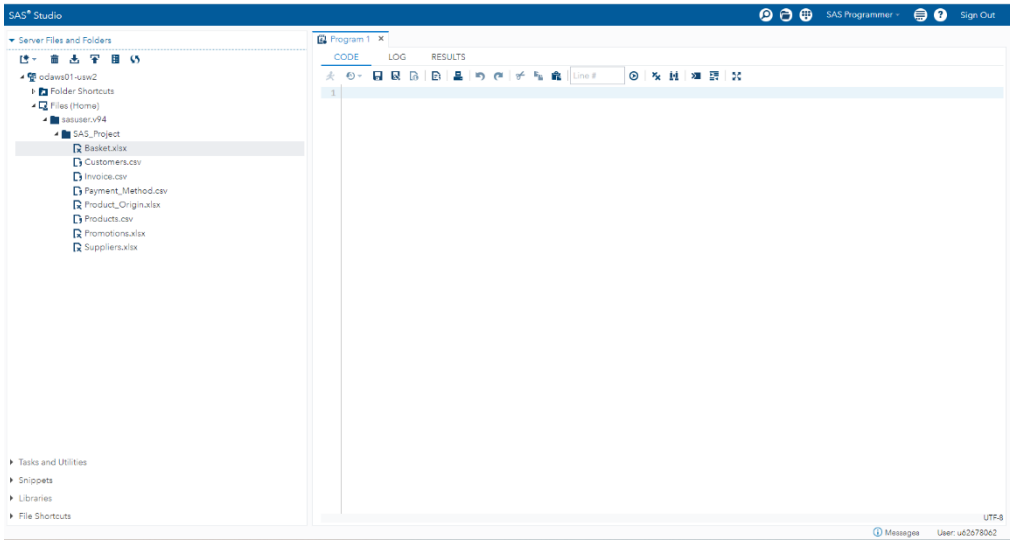

Figure 41 - Data upload

In SAS ODA platform, we firstly uploaded all datasets, as it is shown in Figure 41. We then created a SAS Library, named PROJECT, to import all datasets in SAS format, as it is shown in Figure 42. Then, we edited the Autoexec file, as shown in Figure 43, to maintain the library after logging out from the platform and finally we imported all datasets in SAS format. The SAS code for the data upload and transformations done are included in Figure 44 for SAS ODA and in Figure 45 for SAS Viya (when the errors were fixed). For ease of use, to have a one-picture-for-all, we created a diagram of all SAS datasets and relationships between their variables, like an Entity-Relationship Diagram, which can be viewed in Figure 46.

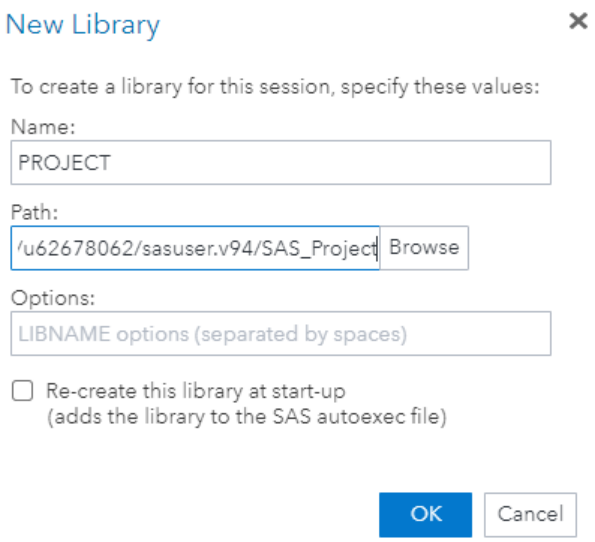

Figure 42 - SAS Library creation

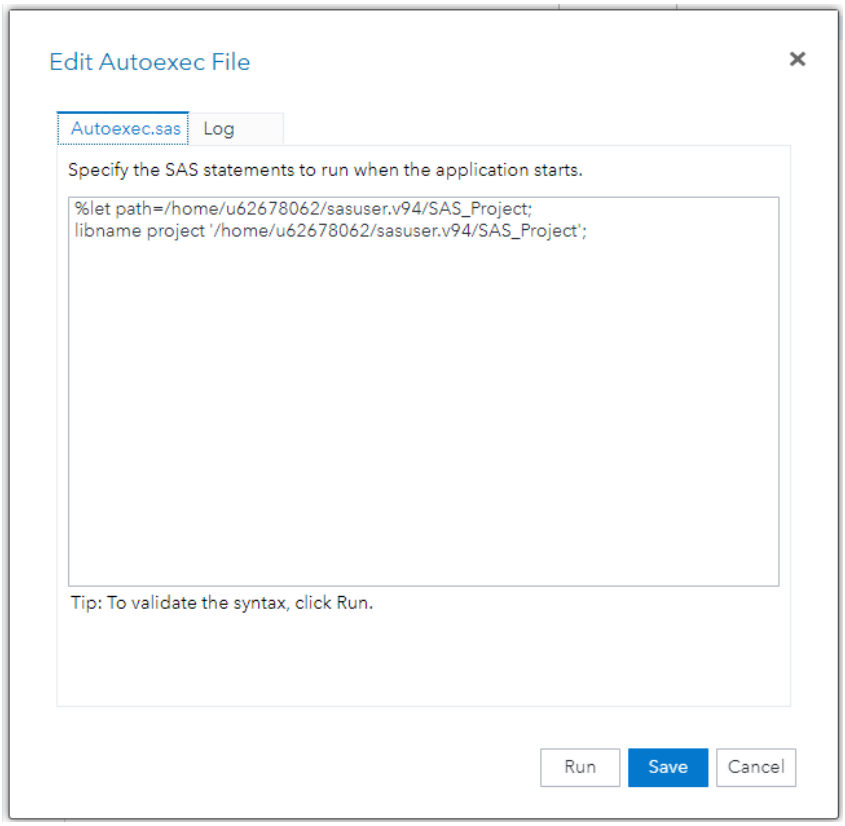

Figure 43 - Edit Autoexec file

```
/*****************************************************************************
* Basket
*****************************************************************************/
%web_drop_table(PROJECT.BASKET);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Basket.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.BASKET;
        GETNAMES=YES;
RUN;
data project.basket;
        set project.basket;
        if not missing(Invoice_ID) and not missing(Product_ID) and not
                missing(Promotion_ID) and not missing(Quantity);
        temp1=put(Invoice_ID, best5.);
        temp2=put(Product_ID, best3.);
        temp3=put(Promotion_ID, best1.);
        drop Invoice_ID Product_ID Promotion_ID;
        rename temp1=Invoice_ID temp2=Product_ID temp3=Promotion_ID;
run;
/*****************************************************************************
* Customers
*****************************************************************************/
%web_drop_table(PROJECT.CUSTOMERS);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Customers.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=CSV OUT=PROJECT.CUSTOMERS;
        GETNAMES=YES;
        GUESSINGROWS = MAX;
RUN;
data project.Customers;
        set project.Customers;
        temp1=put(Customer_ID, best5.);
        temp2=put(Postal_Code, best5.);
        drop Customer_ID Postal_Code;
        rename temp1=Customer_ID temp2=Postal_Code;
run;
/*****************************************************************************
* Invoice
*****************************************************************************/
%web_drop_table(PROJECT.INVOICE);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Invoice.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=DLM OUT=PROJECT.INVOICE;
        DELIMITER=";";
        GETNAMES=YES;
        GUESSINGROWS = MAX;
RUN;
data project.Invoice;
        set project.Invoice;
        temp1=put(Invoice_ID, best5.);
        temp2=put(Customer_ID, best5.);
        temp3=put(Payment_Method, best1.);
        drop Invoice_ID Customer_ID Payment_Method;
        rename temp1=Invoice_ID temp2=Customer_ID temp3=Payment_Method;
run;
/*****************************************************************************
* Payment_Method
*****************************************************************************/
%web_drop_table(PROJECT.payment_method);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Payment_Method.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=DLM OUT=PROJECT.payment_method;
        DELIMITER=";";
        GETNAMES=YES;
        GUESSINGROWS = MAX;
RUN;
data project.Payment_Method;
        set project.Payment_Method;
        temp1=put(Code, best1.);
```

```
/*****************************************************************************
* Product_Origin
*****************************************************************************/
%web_drop_table(PROJECT.PRODUCT_ORIGIN);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Product_Origin.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.PRODUCT_ORIGIN;
	GETNAMES=YES;
RUN;
data project.Product_Origin;
	set project.Product_Origin;
	temp1=put(Code, best1.);
	drop Code;
	rename temp1=Code;
run;
/*****************************************************************************
* Products
*****************************************************************************/
%web_drop_table(PROJECT.Products);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Products.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=CSV OUT=PROJECT.Products;
	GETNAMES=YES;
	GUESSINGROWS = MAX;
RUN;
data project.Products;
	set project.Products;
	temp1=put(Product_ID, best3.);
	temp2=put(SKU, best19.);
	temp3=put(Product_Origin, best1.);
	drop Product_ID SKU Product_Origin;
	rename temp1=Product_ID temp2=SKU temp3=Product_Origin;
run;
/*****************************************************************************
* Promotions
*****************************************************************************/
%web_drop_table(PROJECT.PROMOTIONS);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Promotions.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.PROMOTIONS;
	GETNAMES=YES;
RUN;
data project.Promotions;
	set project.Promotions;
	temp1=put(Promotion_ID, best1.);
	drop Promotion_ID;
	rename temp1=Promotion_ID;
run;
/*****************************************************************************
* Suppliers
*****************************************************************************/
%web_drop_table(PROJECT.SUPPLIERS);
FILENAME REFFILE '/home/u62678062/sasuser.v94/SAS_Project/Suppliers.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.SUPPLIERS;
	GETNAMES=YES;
RUN;
data project.Suppliers;
	set project.Suppliers;
	temp1=put(Supplier_ID, best1.);
	drop Supplier_ID;
	rename temp1=Supplier_ID;
```

Figure 44 - SAS code for importing data on SAS ODA

```
/*****************************************************************************
* Basket
*****************************************************************************/
proc sql;
	%if %sysfunc(exist(PROJECT.BASKET)) %then %do;
	drop table PROJECT.BASKET;
	%end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Basket.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.BASKET;
	GETNAMES=YES;
RUN;
data project.basket;
	set project.basket;
	if not missing(Invoice_ID) and not missing(Product_ID) and not
		missing(Promotion_ID) and not missing(Quantity);
	temp1=put(Invoice_ID, best5.);
	temp2=put(Product_ID, best3.);
	temp3=put(Promotion_ID, best1.);
	drop Invoice_ID Product_ID Promotion_ID;
	rename temp1=Invoice_ID temp2=Product_ID temp3=Promotion_ID;
run;
/*****************************************************************************
* Customers
*****************************************************************************/
proc sql;
	%if %sysfunc(exist(PROJECT.CUSTOMERS)) %then %do;
	drop table PROJECT.CUSTOMERS;
	%end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Customers.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=CSV OUT=PROJECT.CUSTOMERS;
	GETNAMES=YES;
	GUESSINGROWS = MAX;
RUN;
data project.Customers;
	set project.Customers;
	temp1=put(Customer_ID, best5.);
	temp2=put(Postal_Code, best5.);
	drop Customer_ID Postal_Code;
	rename temp1=Customer_ID temp2=Postal_Code;
run;
/*****************************************************************************
* Invoice
*****************************************************************************/
proc sql;
	%if %sysfunc(exist(PROJECT.INVOICE)) %then %do;
	drop table PROJECT.INVOICE;
	%end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Invoice.csv';
```

```
PROC IMPORT DATAFILE=REFFILE DBMS=DLM OUT=PROJECT.INVOICE;
       DELIMITER=";";
       GETNAMES=YES;
       GUESSINGROWS = MAX;
RUN;
data project.Invoice;
       set project.Invoice;
       temp1=put(Invoice_ID, best5.);
       temp2=put(Customer_ID, best5.);
       temp3=put(Payment_Method, best1.);
       drop Invoice_ID Customer_ID Payment_Method;
       rename temp1=Invoice_ID temp2=Customer_ID temp3=Payment_Method;
run;
/*********************************************************************************
* Payment_Method
*********************************************************************************/
proc sql;
       %if %sysfunc(exist(PROJECT.Payment_Method)) %then %do;
       drop table PROJECT.Payment_Method;
       %end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Payment_Method.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=DLM OUT=PROJECT.payment_method;
       DELIMITER=";";
       GETNAMES=YES;
       GUESSINGROWS = MAX;
RUN;
data project.Payment_Method;
       set project.Payment_Method;
       temp1=put(Code, best1.);
       drop Code;
       rename temp1=Code;
run;
/*********************************************************************************
* Product_Origin
*********************************************************************************/
proc sql;
       %if %sysfunc(exist(PROJECT.PRODUCT_ORIGIN)) %then %do;
       drop table PROJECT.PRODUCT_ORIGIN;
       %end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Product_Origin.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.PRODUCT_ORIGIN;
       GETNAMES=YES;
RUN;
data project.Product_Origin;
       set project.Product_Origin;
       temp1=put(Code, best1.);
       drop Code;
       rename temp1=Code;
run;
```

```
/*****************************************************************************
* Products
*****************************************************************************/
proc sql;
	%if %sysfunc(exist(PROJECT.Products)) %then %do;
	drop table PROJECT.Products;
	%end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Products.csv';
PROC IMPORT DATAFILE=REFFILE DBMS=CSV OUT=PROJECT.Products;
	GETNAMES=YES;
	GUESSINGROWS = MAX;
RUN;
data project.Products;
	set project.Products;
	temp1=put(Product_ID, best3.);
	temp2=put(SKU, best19.);
	temp3=put(Product_Origin, best1.);
	drop Product_ID SKU Product_Origin;
	rename temp1=Product_ID temp2=SKU temp3=Product_Origin;
run;
/*****************************************************************************
* Promotions
*****************************************************************************/
proc sql;
	%if %sysfunc(exist(PROJECT.PROMOTIONS)) %then %do;
	drop table PROJECT.PROMOTIONS;
	%end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Promotions.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.PROMOTIONS;
	GETNAMES=YES;
RUN;
data project.Promotions;
	set project.Promotions;
	temp1=put(Promotion_ID, best1.);
	drop Promotion_ID;
	rename temp1=Promotion_ID;
run;
/*****************************************************************************
* Suppliers
*****************************************************************************/
proc sql;
	%if %sysfunc(exist(PROJECT.SUPPLIERS)) %then %do;
	drop table PROJECT.SUPPLIERS;
	%end;
quit;
FILENAME REFFILE DISK
'/shared/home/ele.souflas@aueb.gr/casuser/SAS_Project/Suppliers.xlsx';
PROC IMPORT DATAFILE=REFFILE DBMS=XLSX OUT=PROJECT.SUPPLIERS;
	GETNAMES=YES;
RUN;
data project.Suppliers;
	set project.Suppliers;
	temp1=put(Supplier_ID, best1.);
	drop Supplier_ID;
	rename temp1=Supplier_ID;
run;
```

Figure 45 - SAS code for importing data on SAS Viya

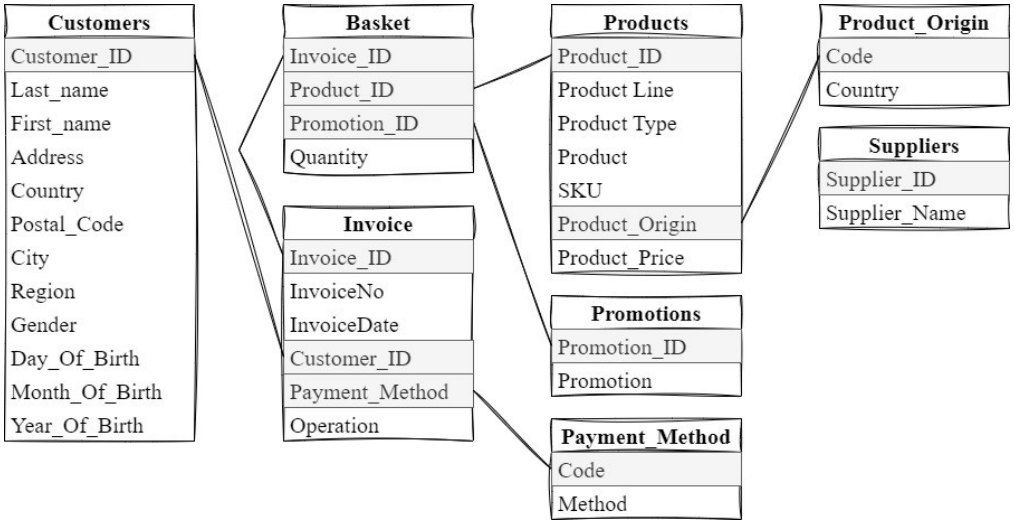

Figure 46 - ER Diagram of the SAS datasets

SAS Code for Data Pre-Processing

All Data Pre-Processing steps are included in the SAS code (Figure 47).

```
/*************************************************************************
 * INVOICE TOTAL ITEMS
 *************************************************************************/
/* Step 1: Merge the relevant datasets */
proc sort data=project.basket out=basket_sorted;
	by invoice_id;
run;
proc sort data=project.invoice out=invoice_sorted;
	by invoice_id;
run;
data basket_invoice;
	merge basket_sorted (in=b) invoice_sorted (in=i);
	by invoice_id;
	if i;
run;
proc sort data=basket_invoice out=basket_invoice_sorted;
	by product_id;
run;
proc sort data=project.products out=products_sorted;
	by product_id;
run;
data basket_invoice_products;
	merge basket_invoice_sorted (in=b) products_sorted (in=p);
	by product_id;
	if b;
run;
/* Step 2: Calculate the number of SKU's for each invoice using PROC SQL */
proc sql noprint;
	create table project.Invoice_Total_Items as select Invoice_ID, COUNT(SKU)
as
		Invoice_Total_Items from basket_invoice_products group by
Invoice_ID;
quit;
/* Step 3: Print the first 10 observations of the new dataset */
proc print data=project.Invoice_Total_Items(obs=10);
run;
```

```
/*********************************************************************************
 * INVOICE TOTAL VALUE
 *********************************************************************************/
/* Step 1: Merge the relevant datasets */
proc sort data=basket_invoice_products
             out=basket_invoice_products_sorted;
      by promotion_id;
run;
proc sort data=project.promotions out=promotions_sorted;
      by promotion_id;
run;
data bask_inv_prod_prom;
      merge basket_invoice_products_sorted (in=a) promotions_sorted (in=b);
      by promotion_id;
      if a;
      /* Calculate the new variable Value_after_discount */
      Value_After_Discount=(1-Promotion)*Product_Price*Quantity;
      /* Format the new variable with two decimal places and no dollar sign */
      format Value_After_Discount COMMA8.2;
run;
/* Step 2: Calculate Invoice_Total_Value using PROC MEANS and OUTPUT statement */
proc means data=Bask_Inv_Prod_Prom noprint nway;
      class Invoice_ID;
      var Value_After_Discount;
      output out=Project.Invoice_Total_Value(drop=_type_ _freq_)
             sum(Value_After_Discount)=Invoice_Total_Value;
run;
/*********************************************************************************
 * INVOICE DIVISION
 *********************************************************************************/
data Project.Sales Project.Returns;
      set Project.Invoice;
      if Operation='Sale' then
             output Project.Sales;
      else if Operation='Return' then
             output Project.Returns;
run;
/*********************************************************************************
 * CUSTOMER'S AGE
 *********************************************************************************/
data Project.Customers;
      set Project.Customers;
      /* Filter valid ages (1910 < age < 2001) */
      where Year_Of_Birth > 1910 and Year_Of_Birth < 2001;
      /* Create a valid birth date using Day, Month, and Year */
      Birth_Date=mdy(Month_Of_Birth, Day_Of_Birth, Year_Of_Birth);
      /* Calculate the age based on the valid birth date
      and adjust for not having reached the birthday for that year */
      Age=floor(intck('year', Birth_Date, '01JAN2019'd) - (Day(Birth_Date) > 1
or
             Month(Birth_Date) > 1));
run;
```

Figure 47 - SAS code for data pre-processing

SAS Code for Customer Profiling

All Customer Profiling steps are included in the SAS code (Figure 48).

```
/*********************************************************************************
 * DEMOGRAPHIC CHARACTERISTICS
 *********************************************************************************/
proc freq data=Project.customers;
   tables Age Gender Region / nocum;
run;
/*********************************************************************************
 * AGE RANGE VARIABLE
 *********************************************************************************/
data project.Customers;
   set Project.Customers;
   format Age_Range $10.; /* Define a custom format for Age_Range */
   if Age < 18 then Age_Range = "Under 18";
   else if Age >= 18 and Age <= 25 then Age_Range = "Very Young";
   else if Age >= 26 and Age <= 35 then Age_Range = "Young";
   else if Age >= 36 and Age <= 50 then Age_Range = "Middle Age";
   else if Age >= 51 and Age <= 65 then Age_Range = "Mature";
   else if Age >= 66 and Age <= 75 then Age_Range = "Old";
   else Age_Range = "Very Old";
run;
/*********************************************************************************
 * BEHAVIORAL CHARACTERISTICS: Visits to the Stores
 *********************************************************************************/
/* Merge relevant datasets */
proc sort data=project.Customers out=Customers_sorted;
       by customer_id;
run;
proc sort data=project.Invoice out=Invoice_sorted;
       by customer_id;
run;
data customers_invoice;
       merge Customers_sorted (in=a) invoice_sorted (in=b);
       by customer_id;
       if a and b;
run;
/* Define a custom format for Age_Range */
proc format;
   value $age_group
         'Under 18' = '1'
      'Very Young' = '2'
      'Young' = '3'
      'Middle Age' = '4'
      'Mature' = '5'
      'Old' = '6'
      'Very Old' = '7';
run;
proc sql noprint;
       create table stores_visits as
       select Age_Range, COUNT(*) as Stores_Visits
       from customers_invoice group by Age_Range;
quit;
/* Apply the custom character format and create a new variable for sorting */
data stores_visits;
   set stores_visits;
   Sort_Order = input(put(Age_Range, $age_group.), $3.);
run;
/* Sort the data by Sort_Order while keeping the original Age_Range values */
proc sort data=stores_visits out=stores_visits;
   by Sort_Order;
run;
```

```
/* Delete the Sort_Order column */
data stores_visits;
   set stores_visits;
   drop Sort_Order;
run;
/*******************************************************************************
 * BEHAVIORAL CHARACTERISTICS: Number of Distinct SKUs purchased
 *******************************************************************************/
/* Merge relevant datasets */
proc sort data=project.customers out=customers_sorted;
       by customer_id;
run;
proc sort data=project.sales out=sales_sorted;
       by customer_id;
run;
data customers_sales;
       merge customers_sorted (in=a) sales_sorted (in=b);
       by customer_id;
       if a and b;
run;
proc sort data=customers_sales out=customers_sales_sorted;
       by invoice_id;
run;
proc sort data=project.basket out=basket_sorted;
       by invoice_id;
run;
data customers_sales_basket;
       merge customers_sales_sorted (in=a) basket_sorted (in=b);
       by invoice_id;
       if a and b;
run;
proc sort data=customers_sales_basket out=customers_sales_basket_sorted;
       by product_id;
run;
proc sort data=project.products out=products_sorted;
       by product_id;
run;
data customers_sales_basket_products;
       merge customers_sales_basket_sorted (in=a) products_sorted (in=b);
       by product_id;
       if a and b;
run;
proc sql noprint;
       create table distinct_SKU as
       select Age_Range, COUNT(DISTINCT SKU) as distinct_SKU
       from customers_sales_basket_products group by Age_Range;
quit;
/* Apply the custom character format and create a new variable for sorting */
data distinct_SKU;
   set distinct_SKU;
   Sort_Order = input(put(Age_Range, $age_group.), $3.);
run;
/* Sort the data by Sort_Order while keeping the original Age_Range values */
proc sort data=distinct_SKU out=distinct_SKU;
   by Sort_Order;
run;
/* Delete the Sort_Order column */
data distinct_SKU;
   set distinct_SKU;
   drop Sort_Order;
run;
```

```
/*********************************************************************************
 * BEHAVIORAL CHARACTERISTICS: Total cost of purchases
 *********************************************************************************/
/* Merge relevant datasets */
proc sort data=customers_sales out=customers_sales_sorted;
      by invoice_id;
run;
proc sort data=project.invoice_total_value out=invoice_total_value_sorted;
      by invoice_id;
run;
data customers_sales_value;
      merge customers_sales_sorted (in=a) invoice_total_value_sorted (in=b);
      by invoice_id;
      if a and b;
run;
proc sql noprint;
      create table total_purchase_cost as
      select Age_Range, SUM(Invoice_Total_Value) as total_purchase_cost
      from customers_sales_value group by Age_Range;
quit;
/* Apply the custom character format and create a new variable for sorting */
data total_purchase_cost;
   set total_purchase_cost;
   Sort_Order = input(put(Age_Range, $age_group.), $3.);
run;
/* Sort the data by Sort_Order while keeping the original Age_Range values */
proc sort data=total_purchase_cost out=total_purchase_cost;
   by Sort_Order;
run;
/* Delete the Sort_Order column */
data total_purchase_cost;
   set total_purchase_cost;
   drop Sort_Order;
   total_purchase_cost=round(total_purchase_cost, 0.01);
run;
/*********************************************************************************
 * PERCENTAGES OF CUSTOMERS IN EACH AGE GROUP
 *********************************************************************************/
/* Frequency table with percentage of customers by age group */
proc freq data=project.Customers noprint;
  tables Age_Range / nocum out=Age_Group_Freq;
run;
/* Apply the custom character format and create a new variable for sorting */
data Age_Group_Freq;
   set Age_Group_Freq;
   Sort_Order = input(put(Age_Range, $age_group.), $3.);
run;
/* Sort the data by Sort_Order while keeping the original Age_Range values */
proc sort data=Age_Group_Freq out=Age_Group_Freq;
   by Sort_Order;
run;
/* Delete the Sort_Order column */
data Age_Group_Freq;
   set Age_Group_Freq;
   drop Sort_Order;
run;
/* Pie chart with percentage of customers by age group */
proc gchart data=Age_Group_Freq;
  pie Age_Range / sumvar=percent;
  title "Percentage of Customers by Age Group";
```

```
/*********************************************************************************
 * BEHAVIORAL CHARACTERISTICS OF EACH AGE GROUP
 *********************************************************************************/
/* Pie chart with visits to the stores by age group */
proc gchart data=stores_visits;
  pie Age_Range / sumvar=stores_visits;
  title "Total Visits to the Stores by Age Group";
/* Pie chart with number of distinct SKUs purchased by age group */
proc gchart data=distinct_SKU;
  pie Age_Range / sumvar=distinct_SKU;
  title "Total Number of Distinct SKUs purchased by Age Group";
/* Pie chart with total cost of purchases by age group */
proc gchart data=total_purchase_cost;
  pie Age_Range / sumvar=total_purchase_cost;
  title "Total Cost of Purchases by Age Group";
```

Figure 48 - SAS code for customer profiling

SAS Code for Sales Exploration

All exploration and understanding of sales steps are included in the SAS code (Figure 49).

```
/*********************************************************************************
 * LEVEL OF SALES AND RETURNS
 *********************************************************************************/
/* Merge the relevant datasets */
proc sort data=project.invoice_total_value out=invoice_value_sorted;
       by invoice_id;
run;
proc sort data=project.invoice out=invoice_sorted;
       by invoice_id;
run;
data sales_returns_value;
       merge invoice_value_sorted (in=a) invoice_sorted (in=b);
       by invoice_id;
       if a and b;
run;
/* Query */
proc sql noprint;
       create table sales_returns_aggr_value as
       select Operation, ROUND(Total_Value, 0.01) as Total_Value from
       (select Operation, SUM(Invoice_Total_Value) as Total_Value
       from sales_returns_value group by Operation);
quit;
/* Create a bar chart for sales and returns */
proc sgplot data=sales_returns_aggr_value;
    vbar Operation / response=Total_Value datalabel;
    xaxis label="Operation";
    yaxis label="Total Value";
    title "Monetary Value by Operation";
run;
/*********************************************************************************
 * AVERAGE BASKET SIZE
 *********************************************************************************/
/* Merge the relevant datasets */
proc sort data=project.invoice_total_items out=invoice_items_sorted;
       by invoice_id;
run;
proc sort data=project.sales out=sales_sorted;
       by invoice_id;
run;
```

```
data sales_items_value;
	merge sales_sorted (in=a) invoice_items_sorted (in=b) invoice_value_sorted
(in=c);
	by invoice_id;
	if a and b and c;
run;
proc sort data=sales_items_value out=sales_items_value_sorted;
	by payment_method;
run;
proc sort data=project.payment_method out=payment_method_sorted;
	by code;
run;
data sales_items_value;
	merge sales_items_value_sorted (in=a) payment_method_sorted (in=b
rename=code=payment_method);
	by payment_method;
	if a and b;
run;
/* Queries */
proc sql noprint;
	create table basket_over_time as
	select Date, ROUND(AVG(Invoice_Total_Items)) as Avg_Items,
	ROUND(AVG(Invoice_Total_Value), 0.01) as Avg_Value from
	(select put(InvoiceDate, YYMMS.) as Date,
	Invoice_Total_Items, Invoice_Total_Value
	from sales_items_value) group by Date;
	create table basket_by_payment as
	select Method as Payment_Method, ROUND(AVG(Invoice_Total_Items)) as Avg_Items,
	ROUND(AVG(Invoice_Total_Value), 0.01) as Avg_Value
	from sales_items_value group by Method;
	select ROUND(AVG(Avg_Items)) into :avg_Avg_Items
    from basket_over_time;
    select ROUND(AVG(Avg_Value),0.01) into :avg_Avg_Value
    from basket_over_time;
quit;
/* Create line charts for average basket size over time */
proc sgplot data=basket_over_time;
    series x=Date y=Avg_Items;
    xaxis label="Month";
    yaxis label="Number of SKUs" min=10 max=20;
    refline &avg_Avg_Items / lineattrs=(pattern=dash)
    label="Average Basket Size:&avg_Avg_Items" labelpos=max;
    title "Average Basket Size Over Time";
run;
proc sgplot data=basket_over_time;
    series x=Date y=Avg_Value;
    xaxis label="Month";
    yaxis label="Basket Monetary Value" min=500 max=1200;
    refline &avg_Avg_Value / lineattrs=(pattern=dash)
    label="Average Basket Value:&avg_Avg_Value" labelpos=max;
    title "Average Basket Monetary Value Over Time";
run;
/* Create bar charts for the average basket size by payment method */
proc sgplot data=basket_by_payment;
    vbar Payment_Method / response=Avg_Items datalabel;
    xaxis label="Payment Method";
    yaxis label="Number of SKUs";
    title "Average Basket Size by Payment Method";
run;
proc sgplot data=basket_by_payment;
    vbar Payment_Method / response=Avg_Value datalabel;
    xaxis label="Payment Method";
    yaxis label="Basket Monetary Value";
    title "Average Basket Monetary Value by Payment Method";
```

```
/*********************************************************************************
 * TOP PRODUCTS
 *********************************************************************************/
/* Merge relevant datasets */
proc sort data=sales_items_value out=sales_items_value_sorted;
       by invoice_id;
run;
proc sort data=project.basket out=basket_sorted;
       by invoice_id;
run;
data basket_sales_value;
       merge sales_items_value_sorted (in=a) basket_sorted (in=b);
       by invoice_id;
       if a and b;
run;
proc sort data=basket_sales_value out=basket_sales_value_sorted;
       by product_id;
run;
proc sort data=project.products out=products_sorted;
       by product_id;
run;
data basket_products_sales_value;
       merge basket_sales_value_sorted (in=a) products_sorted (in=b);
       by product_id;
       if a and b;
run;
/* Report for top products and subtotal sales by product type */
proc sql;
   title "Top Products per Product Line and Product Type";
   SELECT a.'Product Line'n, a.'Product Type'n, Subtotal_Sales, Product as
Top_Product,
   Product_ID as Top_Product_ID, SKU as Top_Product_SKU, Product_Sales as
Top_Product_Sales
   FROM (select 'Product Line'n, 'Product Type'n, c.Product_ID, Product,
       SKU, SUM(Quantity) as Product_Sales
       from project.sales a, project.basket b, project.products c
       where a.Invoice_ID = b.Invoice_ID and b.product_ID = c.product_ID
       group by 'Product Line'n, 'Product Type'n, c.Product_ID, Product, SKU) a,
       (select 'Product Line'n, 'Product Type'n, sum(Quantity) as Subtotal_Sales
    from basket_products_sales_value a
    group by 'Product Line'n, 'Product Type'n) b
   WHERE a.'Product Line'n = b.'Product Line'n AND a.'Product Type'n = b.'Product
Type'n
   GROUP BY a.'Product Line'n, a.'Product Type'n
   HAVING MAX(Product_Sales) = Product_Sales
   UNION
   SELECT 'Product Line'n, '~~~SUM~~~', SUM(Quantity), '~~~', '~~~', '~~~', .
   FROM project.sales a, project.basket b, project.products c
   WHERE a.Invoice_ID = b.Invoice_ID and b.product_ID = c.product_ID
   GROUP BY 'Product Line'n;
quit;
/*********************************************************************************
 * REVENUES BY REGION
 *********************************************************************************/
/* Merge relevant datasets */
/* Customers, Sales, Invoice_Total_Value */
proc sort data=project.customers out=customers_sorted;
       by customer_id;
run;
```

```
proc sort data=project.sales out=sales_sorted;
      by customer_id;
run;
data customers_sales;
      merge customers_sorted (in=a) sales_sorted (in=b);
      by customer_id;
      if a and b;
proc sort data=customers_sales out=customers_sales_sorted;
      by invoice_id;
run;
proc sort data=project.Invoice_Total_Value out=Invoice_Total_Value_sorted;
      by invoice_id;
run;
data customers_sales_value;
      merge customers_sales_sorted (in=a) Invoice_Total_Value_sorted (in=b);
      by invoice_id;
      if a and b;
/* Query */
proc sql noprint;
 CREATE TABLE Region_Revenues as
      SELECT Region, SUM(Invoice_Total_Value) as Revenues
      FROM customers_sales_value
      GROUP BY Region;
 CREATE TABLE Date_Region_Revenues as
      SELECT put(InvoiceDate, YYMMS.) as Date, Region, SUM(Invoice_Total_Value)
as Revenues
      FROM customers_sales_value
      GROUP BY put(InvoiceDate, YYMMS.), Region;
quit;
/* Graphs */
/* Pie chart */
proc gchart data=Region_Revenues;
  pie Region / sumvar=Revenues;
  title "Revenues Contribution of each Region";
run;
/* Create lines chart for revenues per region over time */
proc sort data=Date_Region_Revenues;
    by Date;
run;
proc sgplot data=Date_Region_Revenues;
    series x=Date y=Revenues / group=Region datalabel=Region;
    xaxis label="Month";
    yaxis label="Revenues";
    title "Revenues per Region Over Time";
run;
/*****************************************************************************
 * TOP REGION BY GENDER
 *****************************************************************************/
/* Query */
proc sql noprint;
 CREATE TABLE Gender_SP_Revenues as
      SELECT Gender, Region, SUM(Invoice_Total_Value) as Revenues
      FROM customers_sales_value
      WHERE Region = 'SP'
      GROUP BY Gender, Region
      ORDER BY Gender DESC;
/* Pie chart */
proc gchart data=Gender_SP_Revenues;
  pie Gender / sumvar=Revenues;
  title "Revenues Contribution of SP by Gender";
```

Figure 49 - SAS code for sales exploration

SAS Code for Analysis of Promotional Activities

All promotional activities analysis steps are included in the SAS code (Figure 50).

```
/*************************************************************************
 * PERCENTAGE OF PRODUCTS PER PROMOTION EXISTENCE OR NOT
 *************************************************************************/
/* Create a format */
proc format;
   value PromotionFormatFlag
   0 = 'No Promotion'
   0.1 = 'Promotion'
   0.2 = 'Promotion'
   0.3 = 'Promotion';
run;
/* Merge the relevant datasets */
proc sort data=project.promotions out=promotions_sorted;
       by promotion_id;
run;
proc sort data=project.basket out=basket_sorted;
       by promotion_id;
run;
data basket_promotions;
       merge promotions_sorted (in=a) basket_sorted (in=b);
       by promotion_id;
       if a and b;
run;
/* Frequency table with percentage of products by promotion */
proc freq data=basket_promotions noprint;
  tables Promotion / nocum out=Promotion_Freq;
  format Promotion PromotionFormatFlag.;
run;
/* Pie plot */
proc gchart data=Promotion_Freq;
       pie3d Promotion / sumvar=count clockwise discrete slice=outside
value=outside percent=inside angle=23;
       format Promotion PromotionFormatFlag.;
       format count comma7.;
       pattern1 color=aquamarine;
       pattern2 color=lightgoldenrodyellow;
       pattern3 color=orange;
       pattern4 color=skyblue;
       title "Percentage of Products Sold with/without Promotion";
/*************************************************************************
 * PERCENTAGE OF PRODUCTS PER PROMOTION TYPE
 *************************************************************************/
/* Create a format */
proc format;
   value PromotionFormat
   0 = 'No Promotion'
   0.1 = 'Promotion 10%'
   0.2 = 'Promotion 20%'
   0.3 = 'Promotion 30%';
run;
/* Frequency table with percentage of products by promotion */
proc freq data=basket_promotions noprint;
  tables Promotion / nocum out=Promotion_Freq;
  format Promotion PromotionFormat.;
run;
```

```
/* Create pie plot */
proc gchart data=Promotion_Freq;
       pie3d Promotion / sumvar=count clockwise discrete slice=outside
value=outside percent=inside angle=23 invisible=0 ;
       format Promotion PromotionFormat.;
       format count comma7.;
       pattern1 color=aquamarine;
       pattern2 color=lightgoldenrodyellow;
       pattern3 color=orange;
       pattern4 color=skyblue;
       title "Percentage of Products Sold on Each Promotion Type";
/*****************************************************************************
 * DISTRIBUTION OF SALES PER DAY OF THE WEEK
 *****************************************************************************/
/* Merge the relevant datasets */
data Sales_Merged;
   merge Project.Sales(in=a) Project.Invoice_Total_Items(in=b);
   by Invoice_ID;
   if a and b;
run;
/* Extract the day of the week using the weekday function */
data Sales_Merged;
   set Sales_Merged;
   SaleDay = weekday(InvoiceDate);
   format SaleDay weekdate9.;
run;
/* Sum of distinct SKUs per weekday */
proc means data=Sales_Merged sum mean noprint nway;
    class SaleDay;
    var Invoice_Total_Items;
    output out=Summary(drop= _type_ _freq_)
       n(Invoice_Total_Items)=Total_Sale_Transactions
       sum(Invoice_Total_Items)=Total_Invoice_Distinct_Items
       mean(Invoice_Total_Items)=Average_Invoice_Distinct_Items;
run;
/* Extract the day of the week using the weekday function */
data Sales_Merged;
       set Sales_Merged;
       SaleDay = weekday(InvoiceDate);
       format SaleDay weekdate9.;
       format Average_Invoice_Distinct_Items comma4.1;
run;
/* Print Report */
proc print data=Summary;
   title "Distribution of Sales per Weekday";
run;
/* Pie chart with sale transactions per weekday */
proc gchart data=Summary;
       donut SaleDay / sumvar=Total_Sale_Transactions clockwise discrete
slice=outside value=outside percent=inside angle=0;
       format Total_Sale_Transactions comma5.;
       pattern1 color=aquamarine;
       pattern2 color=lightgoldenrodyellow;
       pattern3 color=orange;
       pattern4 color=skyblue;
       pattern5 color=chocolate;
       pattern6 color=gray;
       title "Distribution of Sales per Weekday";
/* Bar chart with total invoice distinct items per weekday */
```

```
proc sgplot data=Summary;
    vbar SaleDay / response=Total_Invoice_Distinct_Items datalabel;
       format Total_Invoice_Distinct_Items comma6.;
    xaxis label="Weekday";
    yaxis label="Total Invoice Distinct Items";
       title "Total Invoice Distinct Items per Weekday";
run;
/* Bar chart with weekday's distinct items per invoice */
proc sgplot data=Summary;
    vbar SaleDay / response=Average_Invoice_Distinct_Items datalabel;
       format Average_Invoice_Distinct_Items comma4.1;
    xaxis label="Weekday";
    yaxis label="Distinct items per invoice";
       title "Average Invoice Distinct Items per Weekday";
run;
```

Figure 50 - SAS code for analysis of promotional activities

SAS Code for Analysis of Suppliers

All steps for analysis of suppliers are included in the SAS code (Figure 51) below.

```
/*****************************************************************************
 * PERCENTAGE OF PRODUCTS SOLD BY SUPPLIER
 *****************************************************************************/
/* Supplier_ID column in Products table */
data project.products;
       set project.products;
       supplier_id = substr(SKU, 9, 1);
run;
/* Merge the relevant datasets */
proc sort data=project.products out=products_sorted;
       by supplier_id;
run;
proc sort data=project.suppliers out=suppliers_sorted;
       by supplier_id;
run;
data products_suppliers;
       merge products_sorted (in=a) suppliers_sorted (in=b);
       by supplier_id;
       if a and b;
run;
proc sort data=products_suppliers out=products_suppliers;
       by product_id;
run;
proc sort data=project.basket out=basket_sorted;
       by product_id;
run;
data basket_products_suppliers;
       merge products_suppliers (in=a) basket_sorted (in=b);
       by product_id;
       if a and b;
run;
proc sort data=basket_products_suppliers out=basket_products_suppliers;
       by invoice_id;
run;
proc sort data=project.sales out=sales_sorted;
       by invoice_id;
run;
```

```
data sales_basket_products_suppliers;
	merge basket_products_suppliers (in=a) sales_sorted (in=b);
	by invoice_id;
	if a and b;
run;
/* Sum of products per supplier */
proc freq data=sales_basket_products_suppliers;
	tables Supplier_Name / nocum out=Summary;
	weight Quantity;
	title "Percentage of Products Sold by Each Supplier";
run;
proc gchart data=Summary;
	pie3d Supplier_Name / sumvar=count clockwise discrete slice=outside
value=outside percent=inside;
	format count comma7.;
	title "Percentage of Products Sold by Each Supplier";
/*******************************************************************************
 * REVENUES OF PRODUCTS SOLD BY SUPPLIER
 *******************************************************************************/
/* Step 1: Merge the relevant datasets */
proc sort data=project.basket out=basket_sorted;
	by invoice_id;
run;
proc sort data=project.sales out=sales_sorted;
	by invoice_id;
run;
data basket_sales;
	merge basket_sorted (in=a) sales_sorted (in=b);
	by invoice_id;
	if a and b;
run;
proc sort data=basket_sales out=basket_sales_sorted;
	by product_id;
run;
proc sort data=project.products out=products_sorted;
	by product_id;
run;
data basket_sales_products;
	merge basket_sales_sorted (in=a) products_sorted (in=b);
	by product_id;
	if a and b;
run;
proc sort data=basket_sales_products out=basket_sales_products_sorted;
	by promotion_id;
run;
proc sort data=project.promotions out=promotions_sorted;
	by promotion_id;
run;
data bask_sales_prod_prom;
	merge basket_sales_products_sorted (in=a) promotions_sorted (in=b);
	by promotion_id;
	if a and b;
	/* Calculate the new variable Value_after_discount */
	Value_After_Discount=(1-Promotion)*Product_Price*Quantity;
	/* Format the new variable with two decimal places and no dollar sign */
	format Value_After_Discount COMMA8.2;
run;
proc sort data=bask_sales_prod_prom out=bask_sales_prod_prom;
	by supplier_id;
run;
```

```
proc sort data=project.suppliers out=suppliers_sorted;
	by supplier_id;
run;
data bask_sales_prod_prom_sup;
	merge bask_sales_prod_prom (in=a) suppliers_sorted (in=b);
	by supplier_id;
	if a and b;
run;
/* Step 2: Calculate Supplier_Revenues using PROC MEANS and OUTPUT statement */
proc means data=bask_sales_prod_prom_sup noprint nway;
	class Supplier_Name;
	var Value_After_Discount;
	output out=Supplier_Revenues(drop=_type_ _freq_)
		sum(Value_After_Discount)=Supplier_Revenues;
run;
/* Bar chart with Revenues by Supplier */
proc sgplot data=Supplier_Revenues;
    vbar Supplier_Name / response=Supplier_Revenues datalabel dataskin=gloss
categoryorder=respdesc;
	format Supplier_Revenues dollar15.2;
    xaxis label="Supplier";
    yaxis label="Revenues";
	title "Revenues of Products Sold by Each Supplier";
run;
/* The respective donut chart */
proc gchart data=Supplier_Revenues;
	donut Supplier_Name / sumvar=Supplier_Revenues clockwise discrete
slice=outside value=outside percent=inside descending;
	format Supplier_Revenues dollar15.2;
	title "Distribution of Revenues per Supplier";
/*****************************************************************************
 * TOTAL REVENUE OF THE COMPANY W.R.T. ORIGINS OF PRODUCTS SOLD BY SUPPLIER
 *****************************************************************************/
/* Merge relevant datasets */
proc sort data=bask_sales_prod_prom_sup out=bask_sales_prod_prom_sup;
	by product_origin;
run;
proc sort data=project.product_origin out=product_origin_sorted;
	by code;
run;
data bask_sales_prod_prom_sup_or;
	merge bask_sales_prod_prom_sup (in=a) product_origin_sorted
(rename=(Code=product_origin) in=b);
	by product_origin;
	if a and b;
run;
/* Create a cross-tabulation table using proc tabulate */
proc tabulate data=bask_sales_prod_prom_sup_or;
	class Country Supplier_Name;
	var Value_After_Discount;
	table Country='Country of Origin' all='Total',
		(Supplier_Name='Supplier'
all='Total')*Value_After_Discount='''*(sum='''*f=dollar15.2);
	title "Total Revenue by Product Origin & Supplier";
run;
```

Figure 51 - SAS code for analysis of suppliers

SAS Code for Creation of RFM Data

The SAS code used to create the RFM dataset can be displayed below (Figure 52).

```
/*********************************************************************************
 * RFM Data Creation
 *********************************************************************************/
/* The Product_Price, Quantity, and Promotion variables are used for the
creation of the invoice_total_value dataset */
proc sql;
	CREATE TABLE Project.RFM_Data as (
	SELECT a.Customer_ID,
		intck('WEEK', MAX(InvoiceDate), '16dec2011'd, 'C') as R,
		COUNT(b.Invoice_ID) as F,
		SUM(Invoice_Total_Value) as M,
		-1 as T
	FROM project.customers a
		INNER JOIN project.sales b ON a.customer_id=b.customer_id
		INNER JOIN project.invoice_total_value c ON
b.invoice_id=c.invoice_id
	GROUP BY a.Customer_ID
	);
quit;
proc print data=project.rfm_data(obs=10) noobs;
	var Customer_ID R F M;
	title "Sample of 10 Customers' RFM Data";
run;
```

Figure 52 - SAS code for RFM data creation

SAS VDMML Configurations and SAS Code for RFM Customer Segmentation

Firstly, we made the sas7bdat RFM dataset, created in the previous task, an active Cloud Analytics Services (CAS) dataset, after creating a dummy variable as target variable (needed by the software). Then, we created a new SAS Visual Data Mining and Machine Learning (VDMML) project, as shown in Figure 53. From the project settings menu, we declared the entire dataset as training (Figure 54). Then, we created a pipeline adding to our data node the Clustering as child node, the Segment Profile as child node of the Clustering node and the Save Data as child node of the Segment Profile node, as shown in Figure 55. The resulting dataset from this pipeline was saved in the "casuser" folder and after it was made an active CAS dataset (Figure 56), we run the respective SAS code, as shown in Figure 57.

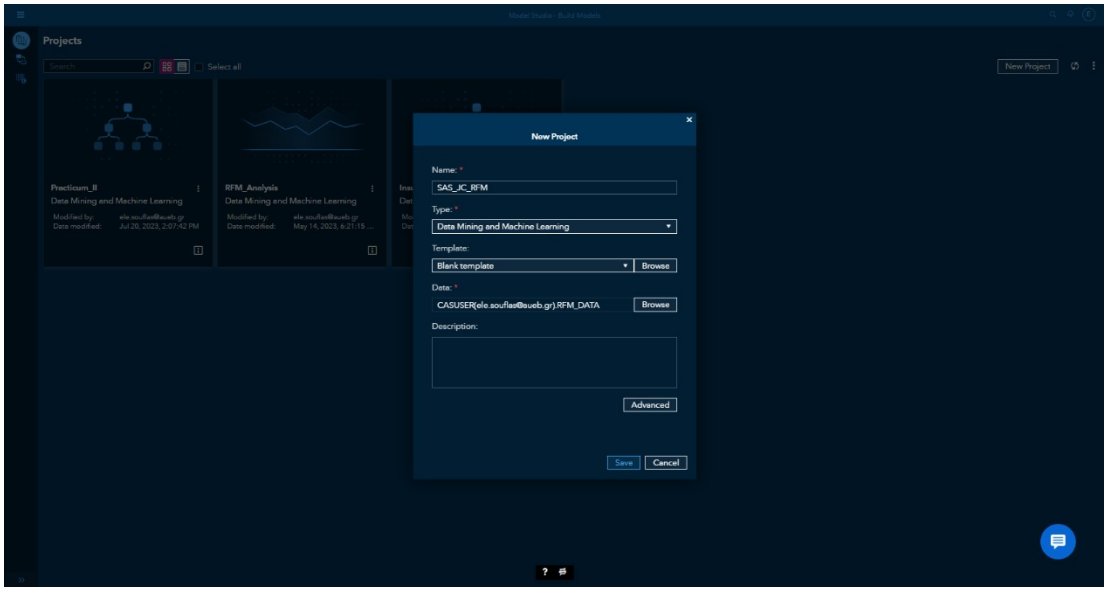

Figure 53 - Creation of new SAS VDMML project

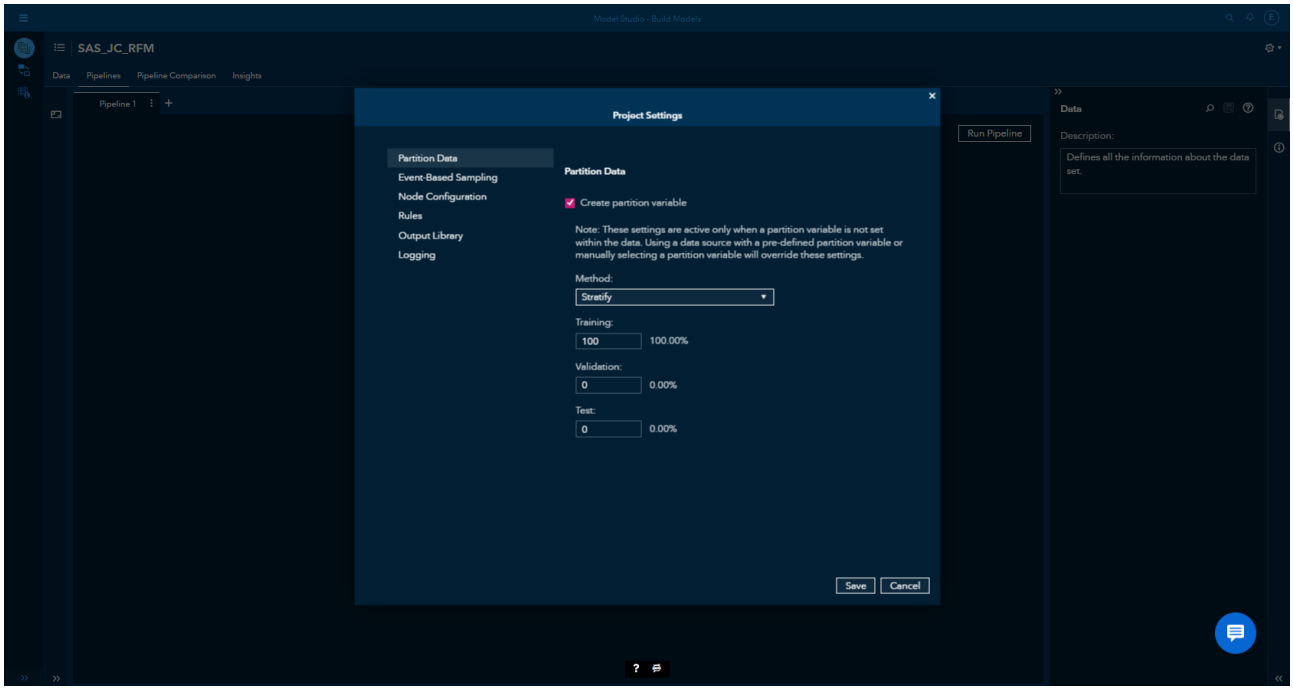

Figure 54 - Declaration of the whole dataset as training

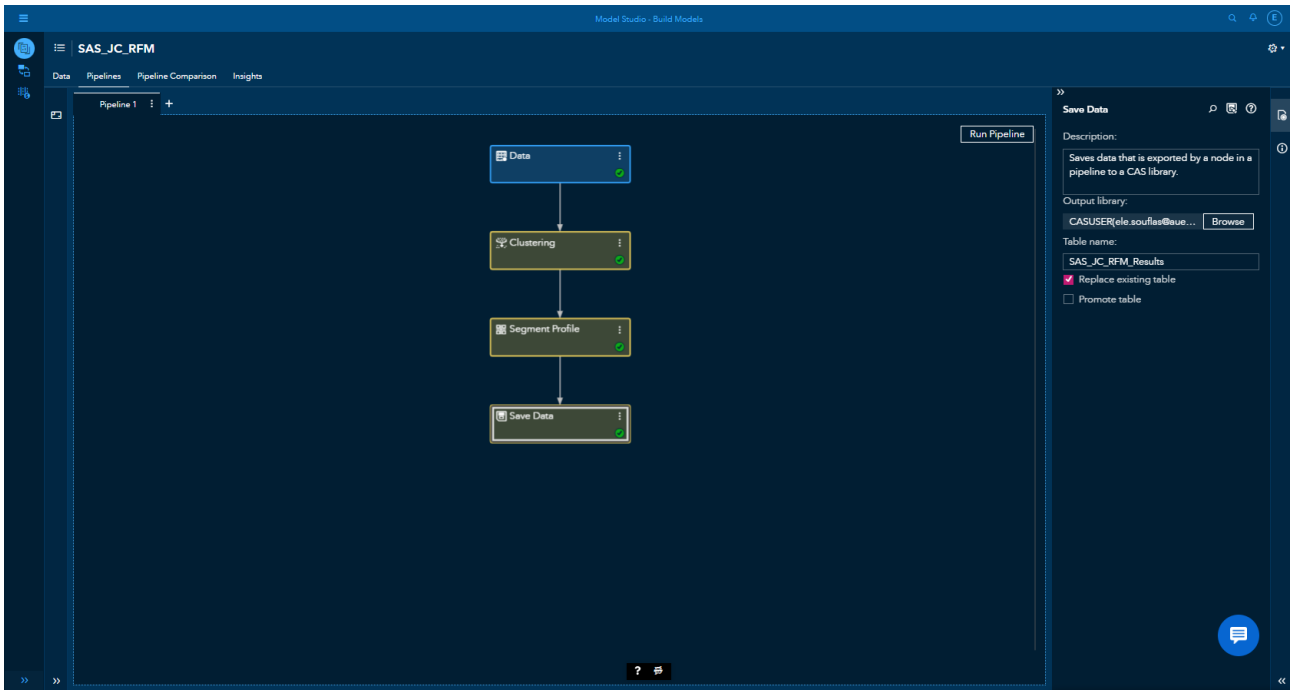

Figure 55 - VDMML Pipeline

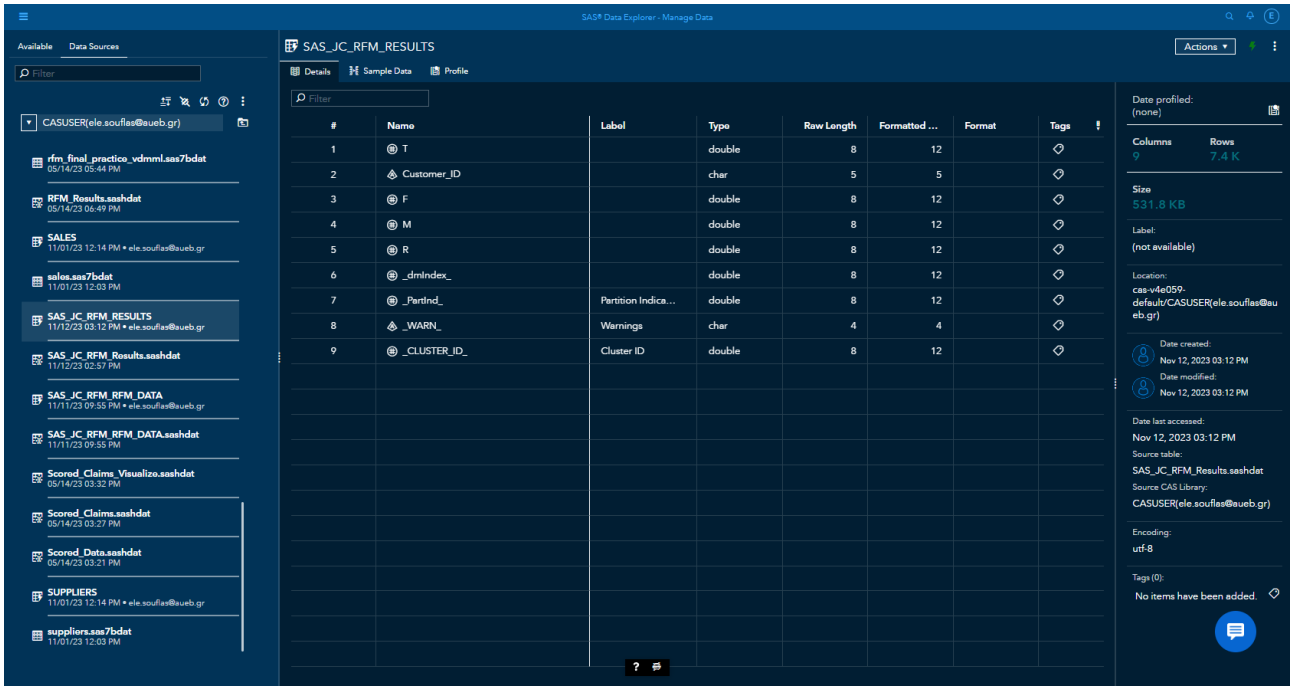

Figure 56 - VDMML Saved Results as active CAS dataset (green thunderbolt)

```
/*****************************************************************************
 * RFM Model Results Explanation
 *****************************************************************************/
/* Convert the CAS Table to SAS Data Set */
libname mycas cas;
data project.RFM_Results;
   set mycas.SAS_JC_RFM_Results (keep = Customer_ID R F M _CLUSTER_ID_);
run;
/* Find the 2 most important clusters created */
title 'Clusters Aggregated Results';
proc sql;
       SELECT _CLUSTER_ID_, COUNT(Customer_ID) as Population,
              MEDIAN(R) as Average_Recency, MEDIAN(F) as Average_Frequency,
              ROUND(AVG(M),0.01) as Average_Monetary
       FROM project.rfm_results
       GROUP BY _CLUSTER_ID_;
quit;
/* Find the customers that are clustered in the most important clusters */
proc sql noprint;
       CREATE TABLE project.cluster_1st_importance as (
       SELECT * FROM project.customers WHERE Customer_ID IN (
       SELECT Customer_ID
       FROM project.RFM_Results
       WHERE _CLUSTER_ID_ = 1));
       CREATE TABLE project.cluster_2nd_importance as (
       SELECT * FROM project.customers WHERE Customer_ID IN (
       SELECT Customer_ID
       FROM project.RFM_Results
       WHERE _CLUSTER_ID_ = 5));
quit;
/*****************************************************************************
 * DEMOGRAPHIC CHARACTERISTICS
 *****************************************************************************/
/* Age */
```

```
proc sgplot data=Project.cluster_1st_importance;
	vbar Age_Range;
	xaxis display=(nolabel) values=('Very Young' 'Young' 'Middle Age' 'Mature'
'Old' 'Very Old');
	title "Distribution of Age in the 1st cluster";
run;
proc sgplot data=Project.cluster_2nd_importance;
	vbar Age_Range;
	xaxis display=(nolabel) values=('Very Young' 'Young' 'Middle Age' 'Mature'
'Old' 'Very Old');
	title "Distribution of Age in the 5th cluster";
run;
/* Gender */
proc gchart data=Project.cluster_1st_importance;
	donut Gender / clockwise discrete slice=outside value=outside
percent=inside;
	title "Distribution of Gender in the 1st cluster";
proc gchart data=Project.cluster_2nd_importance;
	donut Gender / clockwise discrete slice=outside value=outside
percent=inside;
	title "Distribution of Gender in the 5th cluster";
/* Residence */
proc sgplot data=Project.cluster_1st_importance;
	vbar Region;
	xaxis display=(nolabel);
	title "Distribution of Region of Residence in the 1st cluster";
run;
proc sgplot data=Project.cluster_2nd_importance;
	vbar Region;
	xaxis display=(nolabel);
	title "Distribution of Region of Residence in the 5th cluster";
run;
/*****************************************************************************
 * BEHAVIOURAL CHARACTERISTICS
 *****************************************************************************/

proc sort data=project.rfm_results;
	by _CLUSTER_ID_;
run;
proc boxplot data=project.rfm_results(where=(_CLUSTER_ID_ in (1,5)));
	plot R*_CLUSTER_ID_;
	title 'Box Plot for Recency Value';
run;
proc boxplot data=project.rfm_results(where=(_CLUSTER_ID_ in (1,5)));
	plot F*_CLUSTER_ID_;
	title 'Box Plot for Frequency Value';
run;
proc boxplot data=project.rfm_results(where=(_CLUSTER_ID_ in (1,5)));
	plot M*_CLUSTER_ID_;
	title 'Box Plot for Monetary Value';
run;
```

Figure 57 - SAS code for RFM Segmentation

SAS Code for Market Basket Analysis

The SAS code used to conduct the Market Basket Analysis can be displayed below (Figure 58).

```
/*******************************************************************************
 * Market Basket Analysis in the whole dataset
 *******************************************************************************/
/* Market Basket based only on sales and not on  returns transactions */
/* Identify associations of product categories */
proc sql noprint;
       create table project.market_basket as (
       select invoice_id, 'Product Type'n as Product_Category
       from project.basket a, project.products b
       where a.product_id = b.product_id and
              invoice_id in (select invoice_id from project.sales));
quit;
/* Load market_basket dataset to CAS */
cas casauto;
caslib;
libname mycas cas;
data mycas.market_basket;
   set project.market_basket;
run;
/* Market Basket Analysis Procedure */
ods noproctitle;
proc mbanalysis data=mycas.MARKET_BASKET items=4
              conf=0.05 pctsupport=0.05 lift=1;
       target Product_Category;
       customer Invoice_ID;
       output outrule=mycas.MBA_RESULTS;
run;
/* Convert the CAS Table to SAS Data Set */
data project.MBA_Results;
       set mycas.MBA_RESULTS;
run;
/* Print top 10 product category associations */
proc sort data=project.MBA_Results;
       by descending Lift;
run;
proc print data=project.MBA_Results(obs=10) noobs;
       var Rule Lift;
       title "Top 10 Product Categories Associations";
       format Lift 4.2;
run;
/*******************************************************************************
 * Market Basket Analysis in the two most important clusters
 *******************************************************************************/
proc sql noprint;
       create table project.market_basket_cluster_1 as
       select invoice_id, 'Product Type'n as Product_Category
       from project.basket a, project.products b
       where a.product_id = b.product_id and
              invoice_id in (select invoice_id from project.sales c where
              customer_id in (select customer_id from
project.cluster_1st_importance));
       create table project.market_basket_cluster_2 as
       select invoice_id, 'Product Type'n as Product_Category
       from project.basket a, project.products b
       where a.product_id = b.product_id and
              invoice_id in (select invoice_id from project.sales c where
              customer_id in (select customer_id from
project.cluster_2nd_importance));
quit;
```

```
data mycas.market_basket_cluster_1;
   set project.market_basket_cluster_1;
run;
data mycas.market_basket_cluster_2;
   set project.market_basket_cluster_2;
run;
/* Market Basket Analysis Procedure */
ods noproctitle;
proc mbanalysis data=mycas.market_basket_cluster_1 items=4
              conf=0.05 pctsupport=0.05 lift=1;
       target Product_Category;
       customer Invoice_ID;
       output outrule=mycas.MBA_RESULTS_CLUSTER_1;
run;
ods noproctitle;
proc mbanalysis data=mycas.market_basket_cluster_2 items=4
              conf=0.05 pctsupport=0.05 lift=1;
       target Product_Category;
       customer Invoice_ID;
       output outrule=mycas.MBA_RESULTS_CLUSTER_2;
run;
/* Convert the CAS Table to SAS Data Set */
data project.MBA_Results_cluster_1;
       set mycas.MBA_RESULTS_CLUSTER_1;
run;
data project.MBA_Results_cluster_2;
       set mycas.MBA_RESULTS_CLUSTER_2;
run;
/* Print top 10 product category associations for the 1st cluster */
proc sort data=project.MBA_Results_cluster_1;
       by descending Lift;
run;
proc print data=project.MBA_Results_cluster_1(obs=10) noobs;
       var Rule Lift;
       title "Top 10 Product Categories Associations in the 1st Cluster";
       format Lift 4.2;
run;
/* Print top 10 product category associations for the 2nd cluster */
proc sort data=project.MBA_Results_cluster_2;
       by descending Lift;
run;
proc print data=project.MBA_Results_cluster_2(obs=10) noobs;
       var Rule Lift;
       title "Top 10 Product Categories Associations in the 2nd Cluster";
       format Lift 4.2;
run;
```

Figure 58 - SAS code for Market Basket Analysis