REPORT

Data PreProcessing and Visualization

2022/2023

Group H
Catarina Oliveira | 20211616
Inês Vieira | 20211589
Joana Rosa | 20211516
Martim Serra | 20211543

NOVA IMS Information Management School

Table of Contents

Introduction

Nowadays, Data Science is an incredible tool that can be used to improve sales by directing a certain business to the right customers. However, in order to analyse the customer's data, it is essential that the collected data is rightfully treated (data pre-processing).

Given that, *Il Mercato*, a competitive business in the retail market, is looking to get some advantage on its competitors by differentiating themselves on the understanding of their customer's behaviour, it decided to make use of the powers of Data Science.

Therefore, the afore mentioned company provided a group of data scientists with transactional data from the years 2012 to 2014; including both information regarding customer and transactional details, such as: date of birth, nationality, gender, existence of kids; as well as, payment method, product category and subcategory id, transaction date, number of bought items, tax, total amount spent, among others.

Methodology

By working with software, such as:

- SAS Miner for descriptive statistics, treatment of outliers and missing values;
- Python programming language and Excel for data inconsistencies;
- SAS Guide (Programming in Proc SQL) for the creation of new variables;
- And, finally, Power BI for data visualization, the following methodology was applied to *Il Mercato*'s transactional dataset.

Prior to starting the pre-processing of the data, it is important to have a quick look at some initial descriptive statistics in order to check how the raw data is behaving and how it should be treated.

After that, the data must be cleaned, which includes treatment of outliers, imputation of missing values and verification/resolution of any data inconsistencies that might exist among the values of the dataset.

The following step aims to create new variables and transform the ones which already exist with the purpose of reaching more useful insights.

After all the above steps are concluded, the data is finally ready for the creation of an enhanced analytical-based table (ABT) which will allow the company to reach the desired customer segmentation of *Il Mercato*'s clients.

Furthermore, and concerning data visualization, some dashboards will also be created using the already pre-processed data. Therefore, creating a much simpler and quicker way to analyse and get insights of *Il Mercato*'s customers' behaviour.

Descriptive Statistics

Before moving forward to more advanced analysis, and for the purpose of, further on, applying data mining and/or machine learning techniques to this dataset, it is important to recognize some of the patterns in the data.

For clarity reasons, the description of the original variables of the dataset is displayed below, so that it can, at any time, be easily accessed.

Table 1: Description of the variables of the dataset

Variable	Description
transaction_id	Transaction ID
cust_id	Customer ID
tran_dat	Transaction's date
prod_subcat_code	Product subcategory ID
prod_cat_code	Product category ID
Qty	Number of items bought
Tax	Amount of tax paid
total_amt	Amount spent by the customer
Store_type	Sales channel name
Nationality	Customer nationality
Payment_type	Customer payment type
prod_cat	Name of the product category
prod_subcat	Name of the product subcategory
DOB	Customer date of birth
Gender	Customer gender (M/F)
Kids	Customer have kids (1-yes;0-no)
city	Customer city
city_code	City ID
Customer_since	Year of customer first purchase

Table 2: Descriptive Statistics of Numeric Variables of the Dataset

Variable	Median	Missing	Minimum	Maximum	Mean	Standard Deviation
Customer_since	2008	0	2007	2100	2011.95	17.07
total_amt	2072.98	15.0	77.35	4.89E8	74508.94	5533669.12
Tax	196.88	12.0	7.35	7.78E8	192599.28	1.22E7
Qty	3.0	0.0	1.0	5.0	3.00	1.42

Observing table 2, which represents the descriptive statistics from the numeric variables of the dataset, it is possible to conclude that 'Customer_since' clearly has outliers, due to the fact that a customer cannot have made his first purchase in 2100 (extreme value – 85 years in the future, since we are assuming that the present year is 2015).

Given the fact that the maximum total amount spent by a customer is over billions of monetary units (unlikely real-life situation), and that the standard deviation of this feature is also extremely high, 'total_amt' is likely to be a variable with outliers. It can also be concluded, that 'Tax' is a variable which also has outliers, because the maximum value of this feature is even higher than the one from the previously mentioned feature.

Additionally, since 'Qty' is a discrete variable, so none of the above descriptive statistics are the most accurate to perform its characterization. However, the distribution of this feature will be evaluated later.

It is then possible to state that the first three analysed variables ('Customer_since', 'total_amt' and 'Tax') will be the targets for the treatment of outliers of the dataset, which is explained further on in this document.

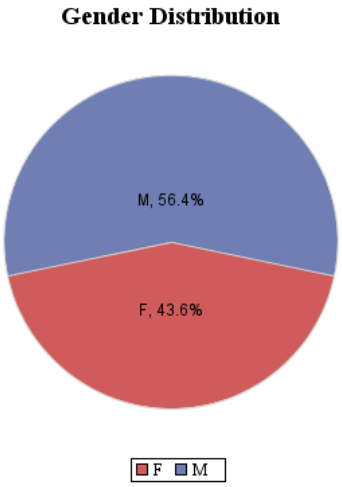
Table 3: Mode and number of missing values of the categorical variables of the dataset

Variable	Missing	Mode
Gender	0	M
Kids	0	1
Nationality	4919	None
Payment_type	0	Credit Card
Store_type	0	Online
city	9	Bilbau
prod_cat	0	Books
prod_subcat	0	Woman
DOB	0	29/04/1972
tran_date	0	23/12/2013

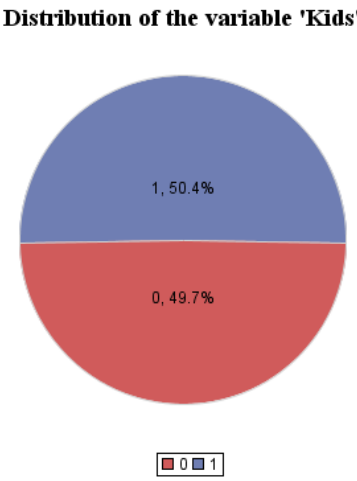
In table 3, it is possible to verify the number of missing values for each categorical variable, as well the mode for each of these features. Some conclusions that can be taken are as follows: 'Nationality' has more missing values than non-null values; therefore, it is also the only categorical variable which does not have a mode; 'city' also has 9 missing values; 29th April 1972 is the date of birth of the customer which made the higher number of transactions in the company, and the day in which most purchases were made is the 23rd December 2013.

Below are, according to the category of their data (demographic data of the customer or transactional data), divisions of certain dataset's variables. This was believed to be an interesting illustration of some of the dataset's variables; therefore, allowing a quicker analysis of these specific customer characteristics.

- Socio-Demographic Variables

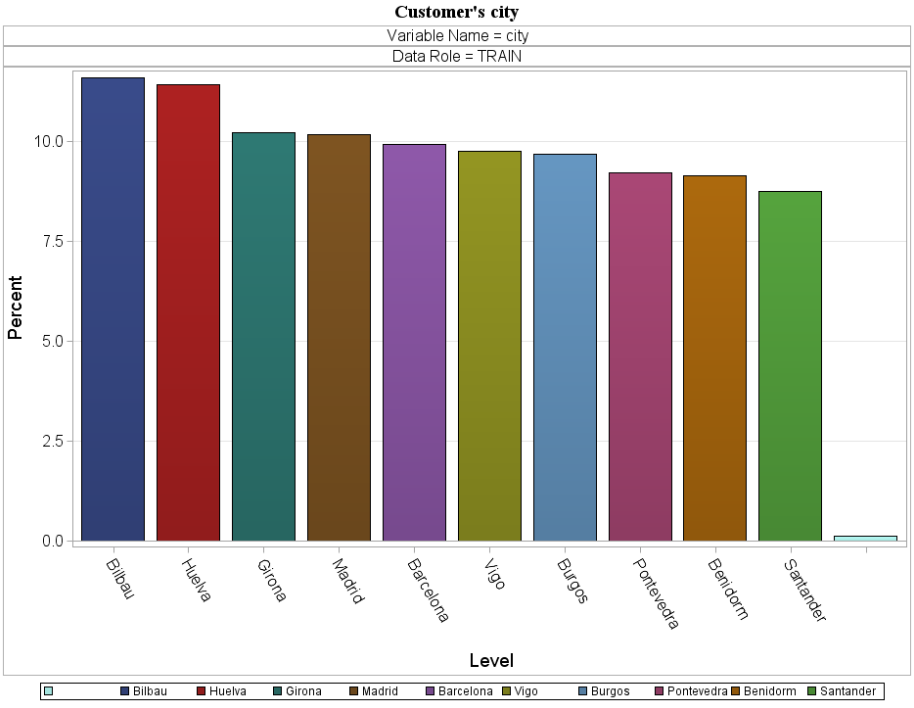


Graphic 1: Customer's Gender Distribution



Graphic 2: Distribution of the variable 'Kids'

It is intriguing to notice that the percentages of the two groups, both in the binary variable 'gender' and 'kids', are quite similar. In short, both groups (male/female and kids/no kids) are identically distributed; such a characteristic improves the reliability of the statistical analysis. Additionally, it can also be concluded that the majority of the customers of *Il Mercato* are males and that most have, at least, one kid.

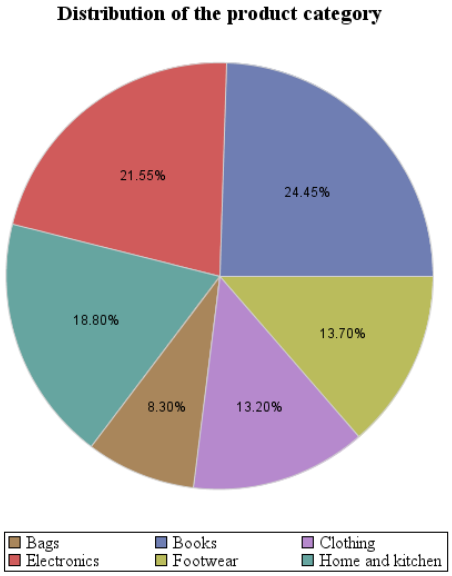


Graphic 3: 'city' distribution

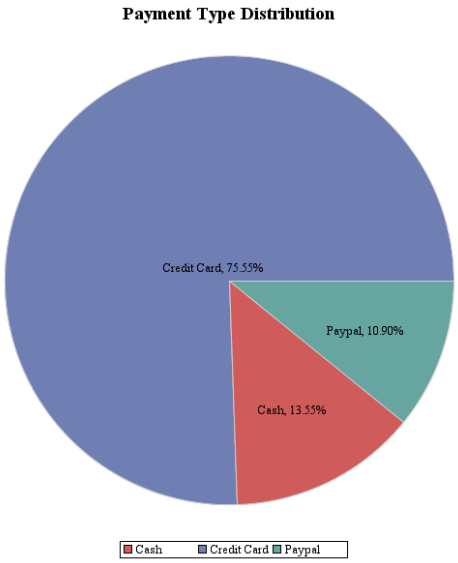
By looking at graphic 3, one can observe that *Bilbau* is the most common city where customers of *Il Mercato* make their purchases and *Santander* is the least common one. Also note that the last bar of the above chart is responsible for representing the missing values, which clearly indicates that the 'city' variable has a small number of them (9 to be exact).

The distribution of the variable 'DOB' (Date of birth) will be shown later in this document since it is severely affected by outliers. Furthermore, the variable 'nationality' is not referenced here, due to the fact that the number of missing values present in this feature was much greater than the one of non-missing values, as it can be seen in table 3. As so, it was already excluded from the dataset.

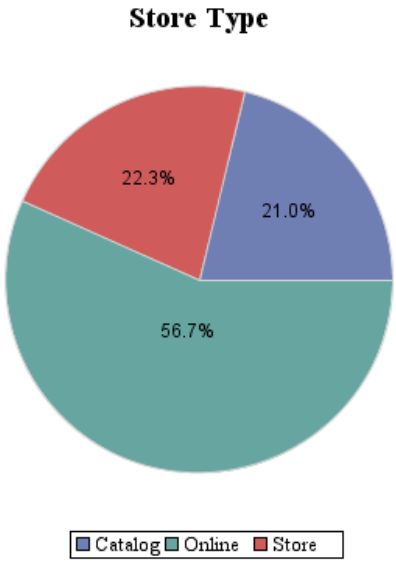
- ## Transaction-specific Variables



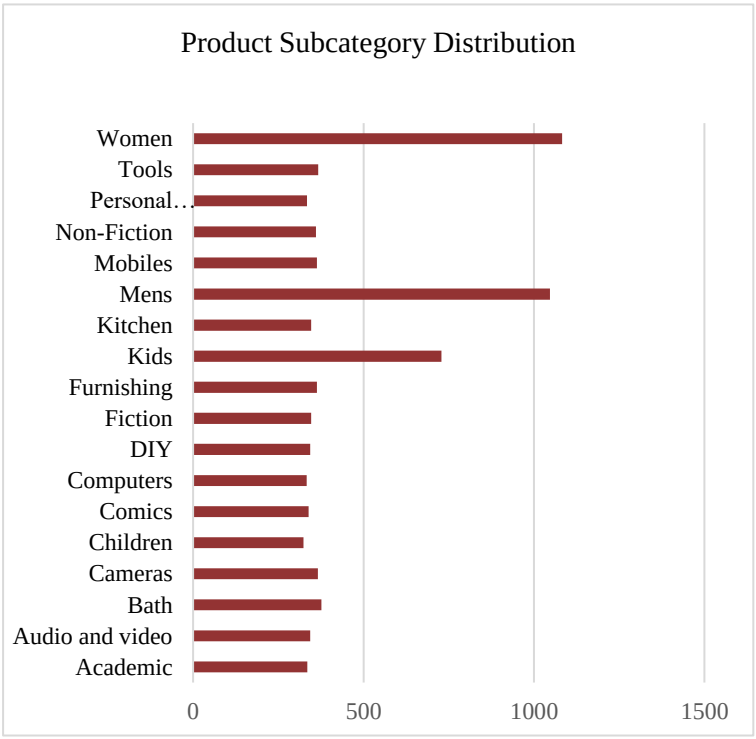
Graphic 4: Product Category Distribution



Graphic 5: Payment Type Distribution



Graphic 6: Distribution of Store Type



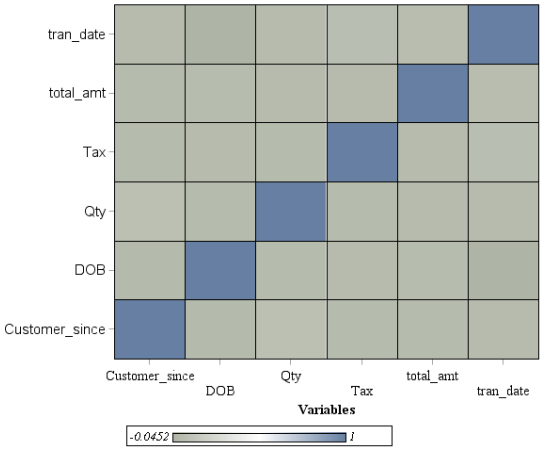
Graphic 7: Distribution of product subcategory

In relation to the variables that are transaction specified, it is interesting to see that Books are the most frequently bought products, followed by electronics (graphic 5). Additionally, Credit Card is, by far, the most common type of payment (graphic 4), which might be related to the fact that the most used Store Type is 'Online' (graphic 6). In Graphic 7, the relationship between the values of

'prod_subcat' was established, and the ones which stood out the most were 'Women', 'Mens', and 'Kids'.

The distributions of the numeric variables of the dataset will, later on, be analysed when detecting the outliers of said dataset features.

Correlation between variables



Graphic 8: Correlation matrix of the interval variables of the dataset

Table 4: Correlation between interval variables of the dataset - with values

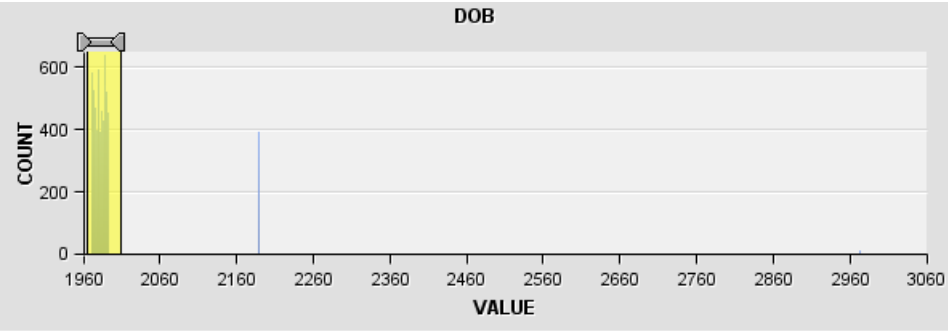
Variable	Variable2	Correlation
Customer_since	Customer_since	1,000
Customer_since	DOB	-0,012
Customer_since	Qty	0,031
Customer_since	Tax	-0,004
Customer_since	total_amt	-0,004
Customer_since	tran_date	-0,001
DOB	Customer_since	-0,012
DOB	DOB	1,000
DOB	Qty	-0,006
DOB	Tax	-0,002
DOB	total_amt	0,002
DOB	tran_date	-0,045
Qty	Customer_since	0,031
Qty	DOB	-0,006
Qty	Qty	1,000
Qty	Tax	-0,006
Qty	total_amt	0,000
Qty	tran_date	-0,001
Tax	Customer_since	-0,004
Tax	DOB	-0,002
Tax	Qty	-0,006
Tax	Tax	1,000
Tax	total_amt	0,000
Tax	tran_date	0,017
total_amt	Customer_since	-0,004
total_amt	DOB	0,002
total_amt	Qty	0,000
total_amt	Tax	0,000
total_amt	total_amt	1,000
total_amt	tran_date	0,013
tran_date	Customer_since	-0,001
tran_date	DOB	-0,045
tran_date	Qty	-0,001
tran_date	Tax	0,017
tran_date	total_amt	0,013
tran_date	tran_date	1,000

When looking at the correlation values between the interval variables of the dataset, it is possible to conclude that none of the variables show a high correlation between each other. Hence, there are not two variables that represent the same information in this dataset.

Treatment of Outliers

Outliers can be defined as extreme values that might represent experimental errors, which can have a significant impact on variable distributions, and, consequently, on the performance of some data mining/machine learning models.

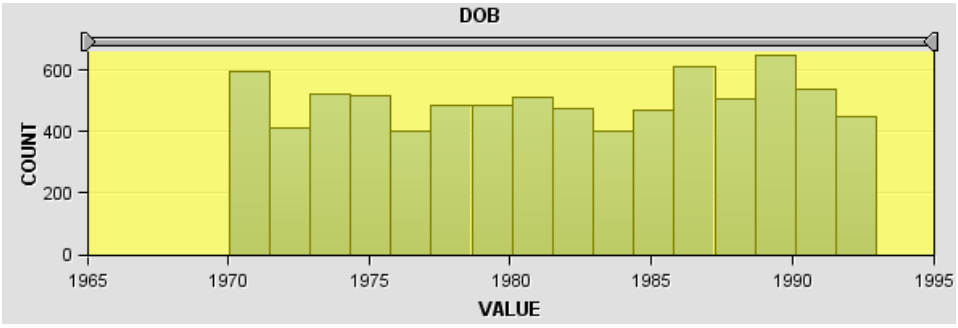
In order to find and treat outliers, the distributions of the interval features of the dataset were observed.



Graphic 9: Distribution of variable 'DOB' before outlier treatment

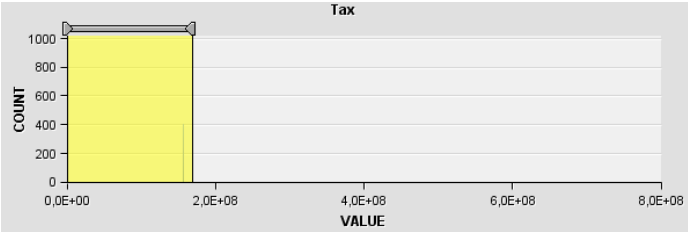
One of the variables that had an extremely skewed distribution was 'DOB' ('Date of Birth'). In graphic 9, it becomes clear that the presence of outliers highly influences the distribution of this dataset's feature. Therefore, these extreme values (including the rest of the rows where these values were inserted) were removed from the dataset using the *filter node* on SAS miner (note that the yellow area represents the values that were kept for further analysis).

The following graphic shows the distribution of the same variable after the outliers were removed (there is no longer the presence of extreme values in this feature).

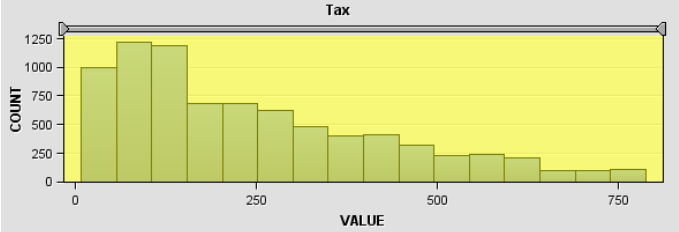


Graphic 10: Distribution of variable 'DOB' after the treatment of outliers

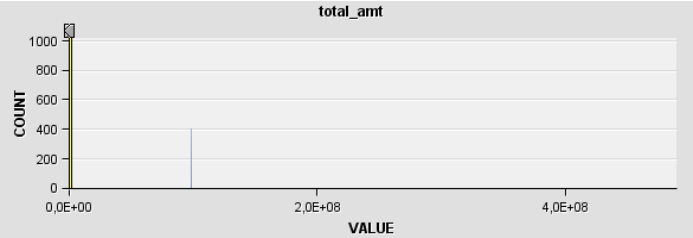
The next few graphics show, as described in the descriptive statistics, that the same exact phenomena happens with other interval variables of the dataset ('tax', 'total_amt' and 'Customer_since').



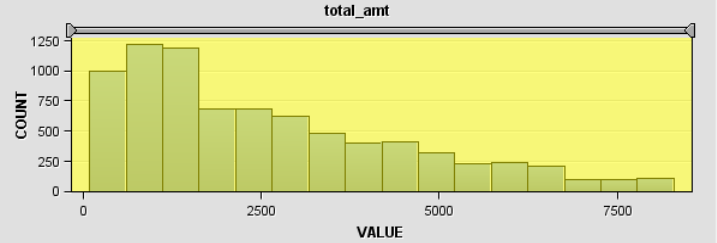
Graphic 11: 'Tax' variable before outlier's treatment



Graphic 12: 'Tax' variable after treatment of outliers

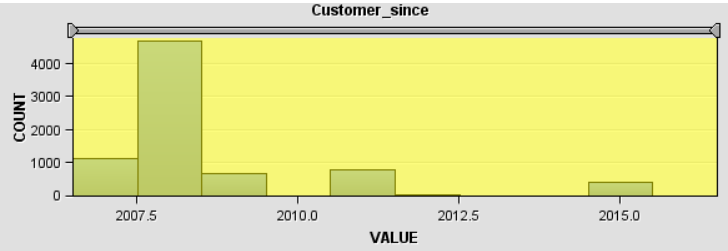


Graphic 13: 'total_amt' variable before outlier's treatment



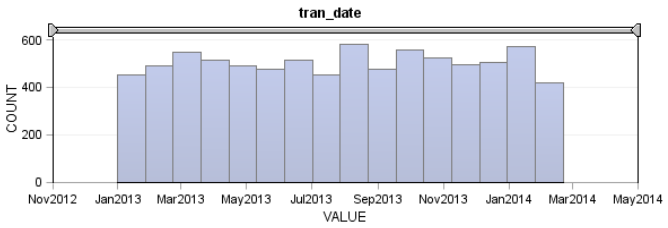
Graphic 14: 'total_amt' after treatment of outliers

Graphic 15: 'Customer_since' before outliers' treatment

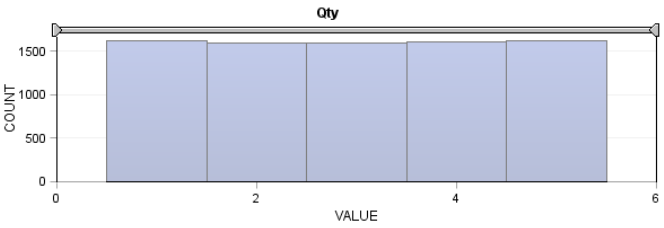


Graphic 16: 'Customer_since' after outliers' treatment

The following graphics represent the distribution of the remaining interval variables of the dataset. These did not need suffer any outlier treatment, since they did not show the presence of extreme values.



Graphic 17: Distribution of 'tran_date'



Graphic 18: Distribution of 'Qty'

After excluding all the above shown outliers, 309 observations were deleted, which represent around 3,75% of the whole dataset. By checking the mean and the median it is now possible to conclude that the deleted outliers of the treated variables no longer affect the first mentioned statistic (it finally looks similar to the median value - table 5).

Table: Repetition of table 1

Variable	Median	Missing	Minimum	Maximum	Mean	Standard Deviation
Customer_since	2008	0	2007	2100	2011.95	17.07
total_amt	2072.98	15.0	77.35	4.89E8	74508.94	5533669.12
Tax	196.88	12.0	7.35	7.78E8	192599.28	1.22E7
Qty	3.0	0.0	1.0	5.0	3.00	1.42

Table 5: Descriptive statistics of numerical variables after treatment of outliers

Variable	Median	Missing	Minimum	Maximum	Mean	Standard Deviation
Tax	195,3	11	7,35	787,5	246,30	187,04
total_amt	2055,3	13	77,35	8287,5	2593,39	1968,84
Qty	3	0	1	5	2,99	1,42
Customer_since	2008	0	2007	2016	2008,68	1,91

Treatment of Missing Values

Due to the disproportionate number of missing values in the variable 'Nationality', and as a way to not impute incorrect data into the dataset, it was opted for its removal from the original Excel file. Even though, it would be great to evaluate, in the visualization part of the project, where most of the customers are from, it would not make sense to do so with inaccurate data, which could possibly ruin further analysis of customer's behaviour.

Furthermore, all missing values from 'city_code' were replaced by the value of the mode (since it is a discrete feature of the dataset), which was equal to 1. Consequently, the missing values in 'city', were imputed with the corresponding city.

The remaining missing values of the variables 'Tax' and 'total_amt' were replaced using the tree method in the *impute node* on SAS Miner.

Total missing values imputed per variable:

- **'city_code'**: 9 missing values (were imputed)
- **'city':** 9 missing values (were imputed)
- '**Tax':** 11 missing values (were imputed)
- **'total_amt':** 14 missing values (were imputed)

Treatment of Data Inconsistencies

It was chosen to treat the data inconsistencies of the dataset using both python and Excel, since it was more practical for the authors to do it with such software, instead of SAS Guide. Nevertheless, the results obtained could easily be achieved using the latter mentioned software.

In sum, the following paragraphs mention the verified variables and which inconsistencies were found in the dataset under study (Note that, for a matter of simplicity, the considered date of analysis was 1st January 2015).

- **Date of birth (DOB)**

One of the found inconsistencies of the dataset concerns 10 rows of the 'DOB' variable. In this case, all these customers had been born in 2974, which is impossible (extreme value 959 years in the future). Therefore, and as these can be considered as both data inconsistencies and outliers, they were excluded from the dataset.

- **Customer_since**

Another found inconsistency was in the variable 'Customer_since', and it is of a similar type as the one in 'DOB', since there were rows that correspond to years in the future: 2100. There were 290 rows with this problem, and as so, they were deleted from the dataset (be aware that some had already been deleted through the treatment of outliers).

- **Customer_since and tran_date**

As it is visible in table 1, 'Customer_since' represents the year of the customer's first purchase on *Il Mercato's* stores, while 'tran_date' denotes the dates of the transactions made by each client.

In 783 rows, two types of inconsistencies regarding 'Customer_since' were found. The first one was that there were cases where the same customer would have more than one 'Customer_since'; as for the second inconsistency it represented the customers which had a higher 'Customer_since' value (year of the first purchase) than the year of their first transaction date within the company (from the provided records of 2012 to 2014).

Therefore, to not lose so much information, for the first case, a series of processes were applied using python programming language, in order to select the earliest 'Customer_since' for each of these "inconsistent" clients. As for the second case, the value of the transaction year was selected and replaced the original 'Customer_since' value.

- **Kids**

The ‘Kids’ variable is no exception when it comes to inconsistencies. In this feature, it was verified that, in the same transaction (checked through ‘transaction_id’), and consequently the same customer (identifiable by the ‘cust_id’), would both have (1) and not have (0) children.

Realistically speaking, this is not a common situation to happen, because people do not just stop being parents in the middle of a transaction. Therefore, given that there was a high number of different clients where this exact situation was verified, to maintain the integrity and quality of the remaining features and observations of the dataset, this column was deleted.

In the next table, which corresponds to a small extract from a larger output, the unique values of the variable ‘Kids’ per ‘transaction_id’ are shown. For instance, the ‘transaction_id’ 141709406 shows two unique values. This indicates that for the same transaction, and consequently the same customer, variable ‘Kids’ shows both 0 and 1 values – data inconsistency.

Table 6: Number of unique values of Kids per transaction id (part of the entire table)

transaction_id	
7073244	1
141709406	2
144983693	1
188992101	2
194354511	2
203539646	2

Once again, this can be considered as an inconsistency of the dataset since one cannot have and not have children at the same time.

- **Minors**

Additionally, it was also verified that there were 124 cases where a minor made a purchase (always considering the date of analysis as 1st January 2015 – these customers were around 15 years old when they became clients of the company). Since this represents an illegal action (minors can only have client cards, depending on the company and country, if they are 16 years old or above), these 124 rows were dropped from the dataset.

Additionally, the following possible data inconsistencies were also verified, but none were found in the dataframe under study.

- The same transactions were all done by the same client;
- All city codes, product categories and subcategories correspond to the same matching value in all observations;
- All customers kept the same gender in each or in the same transaction;
- The same purchase was made through the same store or/and payment types.

Variable Transformation

For the next step of the project, and as already mentioned in the methodology section of this document, variable transformation was conducted using Excel. As so, the following changes were performed:

- **Separation of the 'DOB'** (date of birth) variable into three new others: day of birth (new 'DOB'), month of birth ('MOB') and, finally, year of birth ('YOB');

- **Separation of the 'tran_date'** (date of transaction) into: day ('tran_date_day'), month ('tran_date_month') and year of transaction ('tran_date_year'). Nevertheless, the original 'tran_date' was maintained in the dataset, since it was necessary for the creation of new variables in the next stage of the project.

New Variable Creation

Finally, prior to creating the enhanced analytical-base-table (ABT), new variables were computed (from already existing ones in the dataset), through SAS Guide (Proc SQL).

Below is a list with all newly created variables, as well as considerations in relation to some of them and what they represent.

- per_tax_amt (tax percentage paid, per customer, per transaction)
- age
- first_purchase
- last_purchase
- client_yrs
- avg_amt
- customer_seg
- qty_(category name)
- qty_pay_(Payment type name)
- qty_(Store type name)
- qty_city_(City name)
- freq_(category name)
- freq_pay_(Payment type name)
- freq_channel_(Store type name)
- mon_(category name)
- mon_(subcategory name)
- mon_pay_(Payment type name)
- mon_(Store type name)

Since there were many subcategories in each of the company's products categories it was opted to not proceed with the creation of the frequency and quantity variables in terms of 'prod_subcat'. This is due to the fact that overly specific variables would be generated, and they would not be as relevant for the final analytical-base-table process as other newly created variables.

Also note that the new variable 'per_tax_amt' is a variable which contains similar values - around 9.5%. Therefore, even though it was a helpful feature to understand the amount of taxes paid by the

customers at the company's stores, it is not crucial for it to belong to the final enhanced analytical-base-table (ABT).

Furthermore, the variables 'qty_city_(name of city)' were also created. However, as the majority of the customers only made their purchases in one store, it would not make sense to keep them as relevant variables for customer segmentation (not included in the ABT). Nevertheless, the visualization of this exact phenomena will be explored further on in the Data Visualization section of this document.

Finally, 'avg_amt' which represents the average total amount spent per customer in the company, was originally created for each year ('avg_amt_2012', 'avg_amt_2013','abg_amt_2014'). However, as the year of 2012 does not have many values and the information on customer transactions over the year of 2014 is scarce it was decided that only one combined variable of average amount spent would be presented in the final ABT.

ABT

An Analytical-Base-Table, also known as ABT, is a flat table which holds the necessary information for data mining tools. Furthermore, it defines if the analytical tasks to be performed will be predictive or descriptive. Moreover, a row in this type of table represents the object in study (in this case, a customer – table is aggregated per 'cust_id') and the columns describe the target (client). Overall, the final goal of an ABT is to allow the prediction and analysis of the company customer's behavior.

For this project's ABT, created through SAS Guide (Proc SQL), the most relevant variables of the dataset (both original and newly created ones) were selected and grouped by the customer ID ('cust_id'). In the table below is the list of all considered variables in the final ABT (when there is an original variable name in parentheses in front of the new variable it means that there is a column for each category of such variable).

New Variable	Description
cust_id	Customer ID
City	Customer's city
Gender	Customer's Gender
Date_Birth	Customer's date of birth
age	Age of the customer's in 2015 (date of analysis)
first_purchase	Date of customer's first purchase
last_purchase	Date of customer's last purchase
client_yrs	Number of years, until 2015, since the customer is a client
avg_amt	Average total amount spent per customer
customer_seg	Division of customers per different ranks (Bronze, Silver, Gold, Platinum) according to their total amount spent in the company
qty_(category name)	Quantity of products bough, per customer, per category
qty_pay_(Payment type name)	Quantity of purchases, per customer, paid with a certain payment type
qty_(Store type name)	Quantity of purchases, per customer, by store type
freq_(category name)	Frequency with which a customer bough a product from a certain category
freq_pay_(Payment type name)	Frequency with which a customer paid with a certain payment type
freq_channel_(Store type name)	Frequency with which a customer bough items by different store types
mon_(category name)	Total amount spent, per customer, in each product category
mon_(subcategory name)	Total amount spent, per customer, in each product subcategory
mon_pay_(Payment type name)	Total amount paid, per customer, with a certain payment type
mon__(Store type name)	Total amount spent, per customer, in different store types

Data Visualization

Large datasets have the problem of being difficult to interpret and to get conclusions from. Visualizations allow a better comprehension of the data, identification of patterns and errors and are also effortlessly perceived by human beings; therefore, easing the decision-making process.

Visualizations for *Il Mercato* were conducted on Power BI and are summarized in the following three Dashboards.

The first dashboard focuses on the Custumers' General Information, presenting further insights on the clients of the company. The second and third dashboards describe the general Transactional Behaviour divided in how transactions differ according to some customer information and how they differ given their preferences, respectively.

The entire visualization respected a color pallete composed by the blue used *in Il Mercato* Logo, some tones of grey, white and other pastel colors like baby pink, yellow, and another blue tone.

The Customer's General Information dashboard is out of the three the most interactive one, having several buttons and slicers that filter the information allowing more depthfull insights. These buttons can filter by the data gender and the slicers by age and transaction date. Also notice that the map of Spain is where one can filter information based on the province (of the city) where the customers made their transactions. The other graphics presented in this dashboard are: a histogram, in which each bar tells how many customers there are with a specific age, and a pie chart that shows the percentage of

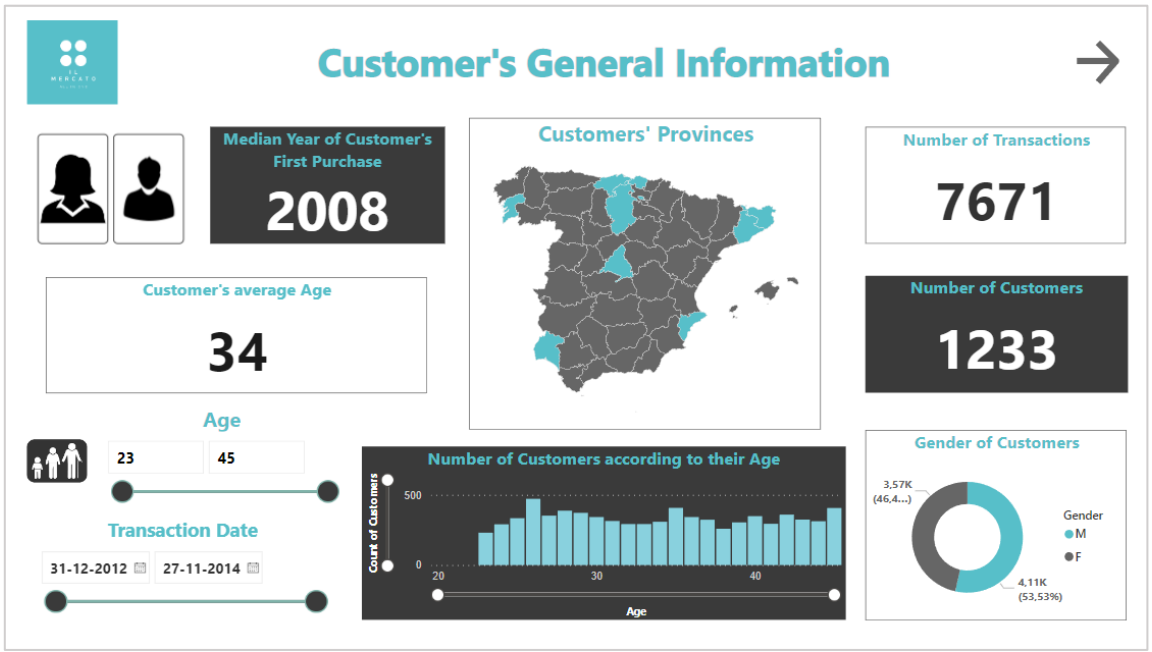


Figure 1: First Dashboard - Customer's General Information

customers according to their gender. This dashboard also includes some more useful information in boxes like the total number of transactions, total number of customers, customers' average age and the median year of customers' first purchase.

The second dashboard, named Transactional Behaviour, like said before, relates information about the customers to their transactional behaviour and is composed by five graphics and one slicer. This slicer, just as in the previous dashboard, filters the data by transaction date. The first graphic on this dashboard is a treemap that represents the proportion of the customers that preferred to buy through a specific sales channel. Others include a pie chart that compares the total amount spent according to the customer's gender, a gauge chart representing the average amount spent per transaction, a funnel graph that shows how the amount spent varies depending on the customer's age and finally, a multiple window line chart that compares the average amount spent per year and month. Notice that this last graph seems to not present any data in 2012, in the following figure. However, that is not exactly true, it is just not visible in the figure due to its staticity and because the transations of this year were all from December. Nevertheless, it is a very useful graph to visualize when customers made most of their transactions and compare them with the amount spent.

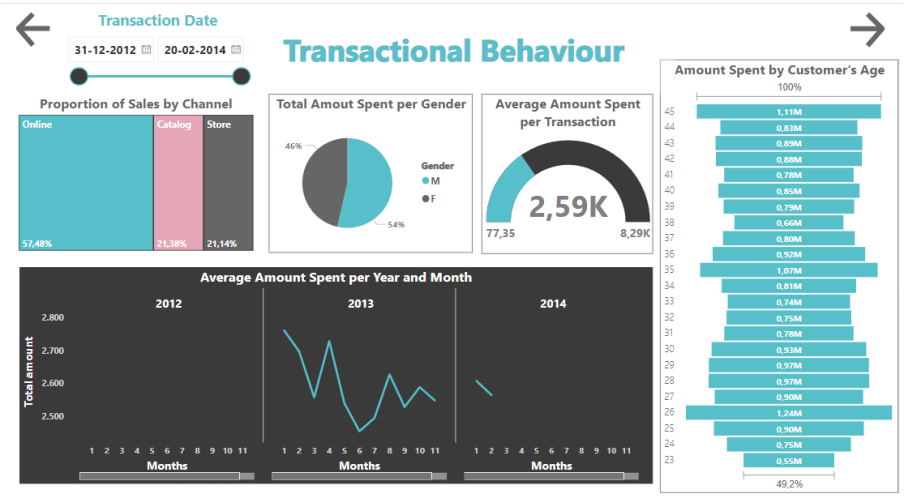


Figure 2: Second Dashboard - Transactional Behaviour

The third dashboard is somewhat of a continuation of the second dashboard and, as so, it is also named Transactional Behaviour. It relates customers' preferences to their transactional behaviour. It is composed by four graphs, one slicer and one box. The slicer is the same as in the second and first dashboards. The box shows the revenue, or in other words the total amount gathered by the sales. There are also two bar charts in this dashboard: the first one represents the percentage of sales (transactions) per Payment Type and the other displays the average amount spent by the customers in each category and subcategory of products. The remaining two charts are a donut chart, in which is

possible to compare the percentage of the number of transactions according to each product category, and a 100% stacked bar chart that shows the percentage of each sales channel according to the product category sold.

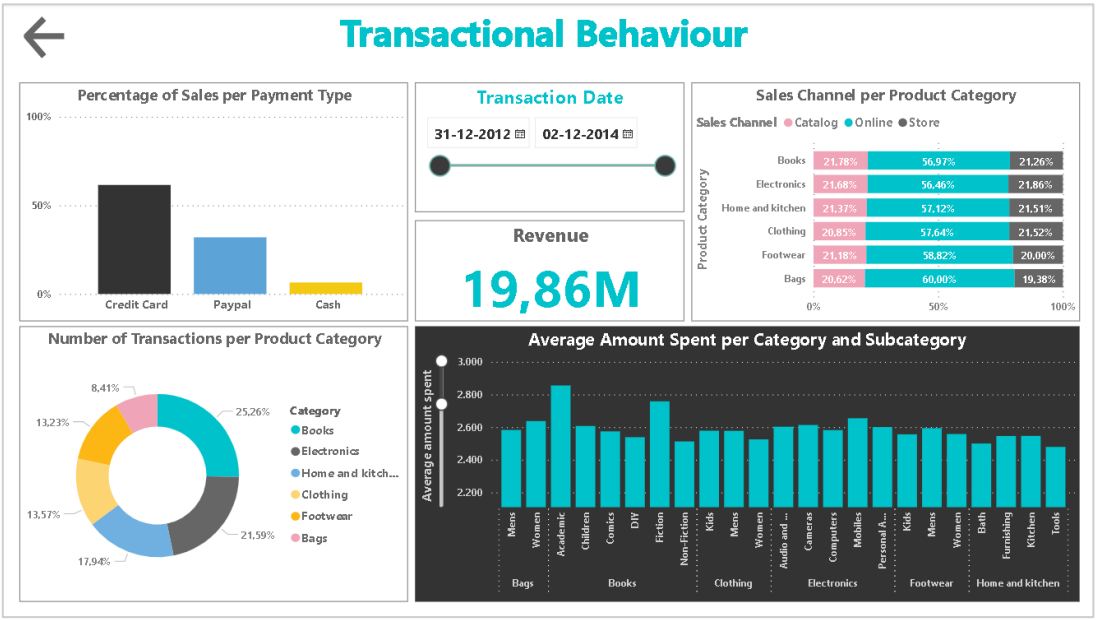


Figure 3 - Third Dashboard - Transactional Behaviour

Conclusion

By preprocessing the data, creating the ABT and the three dashboards it was possible to infer some insights from the provided data, in order to answer simple business questions and segment the customers from *Il Mercato*. The following conclusions were made according to the preprocessed dataset and the created dashboards.

Regarding the company's customers, it was possible to conclude that 53% of the 1233 customers are males and about 47% are females. The average age of the customers is 34 years and they are distributed by 9 out of 50 provinces. In other words, the company is only present in 18% of the provinces in Spain.

Concerning the sales made by the company, 7671 transactions were analysed and a revenue of 19,86 millions was achieved. The total amount spent by the customers was slightly higher for males (54%) than for females (46%) and the customers who spent the largest amount of money were 26 years old.

The preferred payment type, by the customers of *Il Mercato,* in more than 60% of the transactions was the credit card and the favorite store type was Online for all of the product categories.

The most bought product category was books (around 25% of the total transactions) followed by eletronics (21%). Within the book subcategories, the one who had a bigger average amount spent per transaction was Academic books with 2855 monetary units, followed by fiction books (average amount equal to 2758 monetary units). However, the inverse happens in the sum of the total amount meaning that overall more money was spent on fiction books (around 915 000) than academic ones (885 000).

The second most bought category was eletronics and the subcategory that stood out among the others was Mobiles. The average amount spent in this subcategory, per transaction, was 2654 monetary units with the sum of total amount spent being of 915 000, so it was the most sold product and also the one who gathered the biggest amount of money in the category of eletronics.

The category that was less sold were bags (bought in around 8% of the transactions) and it was also the one which generated less revenue (1.68M). Even though the average amount spent per transaction on both subcategories (Men and Women) are not the lowest, the summed amount on men's bags is the third-worst of all subcategories (with a revenue of 810 000).

With all the reliable information and insights provided by the Data Preprocessing team, *Il Mercato* has now the possibility to better understand their customers needs and their behaviour, which will hopefully lead to successful decision-making and the growth of the company.

Annex 1: SAS Miner Diagram

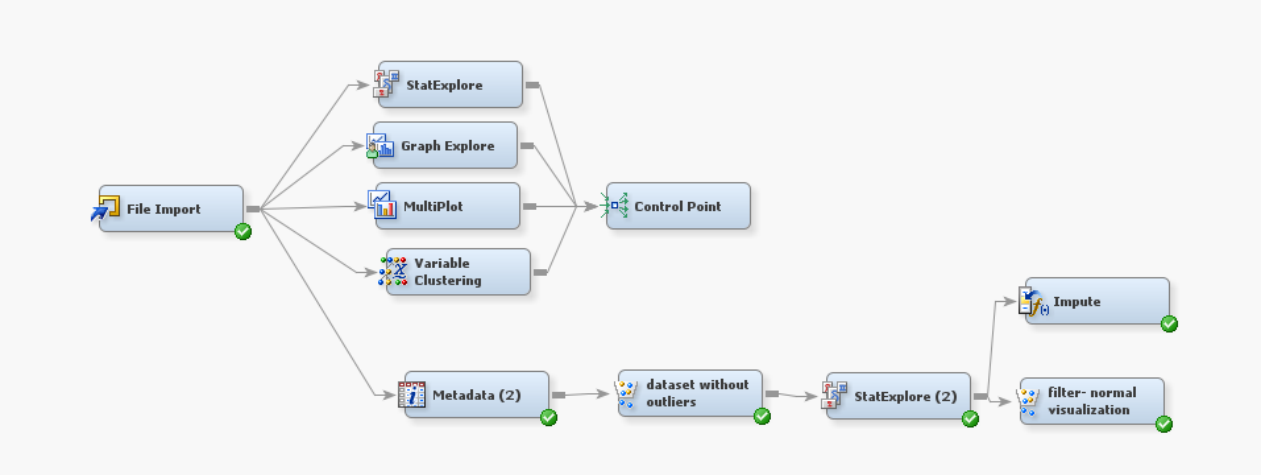


Figure 4: SAS Miner diagram

Anex 2: SAS Guide (Proc SQL) code

```
/*frequency per product categories*/

PROC SQL;
CREATE TABLE frequency as
SELECT DISTINCT cust_id, prod_cat, count(distinct(transaction_id)) as frequency
FROM work.final_data
GROUP BY cust_id, prod_cat;
RUN;

PROC SORT DATA=frequency;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=frequency
OUT=abt_freq_cat
PREFIX=freq_;
ID prod_cat;
BY cust_id;
RUN;
```

```
/*frequency per store type*/

PROC SQL;
CREATE TABLE frequency_sale_chan as
SELECT DISTINCT cust_id, Store_type, count(distinct(transaction_id)) as frequency_sale_chan
FROM work.final_data
GROUP BY cust_id, Store_type;
RUN;

PROC SORT DATA=frequency_sale_chan;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=frequency_sale_chan
OUT=freq_sal_chan
PREFIX=freq_channel_;
ID Store_type;
BY cust_id;
RUN;
```

```
 /*monetary per subcategory*/

PROC SQL;
CREATE TABLE monetary_subcat as
SELECT DISTINCT cust_id, prod_subcat, sum(total_amt) as monetary_subcat
FROM work.final_data
GROUP BY cust_id, prod_subcat;
RUN;

PROC SORT DATA=monetary_subcat;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=monetary_subcat
OUT=abt_mon_subcat
    PREFIX=mon_;
ID prod_subcat;
BY cust_id;
RUN;
```

```
/*frequency per payment type*/

PROC SQL;
CREATE TABLE frequency_pay as
SELECT DISTINCT cust_id, Payment_type, count(distinct(transaction_id)) as frequency_pay
FROM work.final_data
GROUP BY cust_id, Payment_type;
RUN;

PROC SORT DATA=frequency_pay;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=frequency_pay
OUT=abt_freq_pay
PREFIX=freq_pay_;
ID Payment_type;
BY cust_id;
RUN;
```

```
/*monetary per category*/

PROC SQL;
CREATE TABLE monetary as
SELECT DISTINCT cust_id, prod_cat, sum(total_amt) as monetary
FROM work.final_data
GROUP BY cust_id, prod_cat;
RUN;

PROC SORT DATA=monetary;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=monetary
OUT=abt_mon_cat
    PREFIX=mon_;
ID prod_cat;
BY cust_id;
RUN;
```

```
/*monetary per Payment Type*/

PROC SQL;
CREATE TABLE monetary_pay as
SELECT DISTINCT cust_id, Payment_type, sum(total_amt) as monetary_pay
FROM work.final_data
GROUP BY cust_id, Payment_type;
RUN;

PROC SORT DATA=monetary_pay;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=monetary_pay
OUT=abt_mon_pay
    PREFIX=mon_pay_;
ID Payment_type;
BY cust_id;
RUN;
```

```
/*monetary per Store Type*/

PROC SQL;
CREATE TABLE monetary_store as
SELECT DISTINCT cust_id, Store_type, sum(total_amt) as monetary_store
FROM work.final_data
GROUP BY cust_id, Store_type;
RUN;


PROC SORT DATA=monetary_store;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=monetary_store
OUT=abt_mon_store
    PREFIX=mon_;
ID Store_type;
BY cust_id;
RUN;
```

```
/*last_purchase*/

PROC SQL;
CREATE TABLE abt_8 AS
SELECT DISTINCT cust_id, max(tran_date) as last_purchase
FROM work.final_data
GROUP BY cust_id;
RUN;

DATA abt_9;
SET abt_8;
FORMAT last_purchase date9.;
RUN;
```

```
/*%Taxpertotal_amt*/

PROC SQL;
CREATE TABLE abt_13 AS
SELECT DISTINCT cust_id, ((tax/total_amt)*100) as per_tax_amt
FROM work.final_data
GROUP BY cust_id;
RUN;
```

```
/*Quantity of products paid per Payment type*/

PROC SQL;
CREATE TABLE qty_pay as
SELECT DISTINCT cust_id, Payment_type, sum(Qty) as qty_pay
FROM work.final_data
GROUP BY cust_id, Payment_type;
RUN;


PROC SORT DATA=qty_pay;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=qty_pay
OUT=abt_qty_pay
    PREFIX=qty_pay_;
ID Payment_type;
BY cust_id;
RUN;
```

```
/*Quantity of products bought per Store type*/

PROC SQL;
CREATE TABLE qty_store as
SELECT DISTINCT cust_id, Store_type, sum(Qty) as qty_store
FROM work.final_data
GROUP BY cust_id, Store_type;
RUN;


PROC SORT DATA=qty_store;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=qty_store
OUT=abt_qty_store
    PREFIX=qty_;
ID Store_type;
BY cust_id;
RUN;
```

```
/*first_purchase*/

PROC SQL;
CREATE TABLE abt_6 AS
SELECT DISTINCT cust_id, min(tran_date) as first_purchase
FROM work.final_data
GROUP BY cust_id;
RUN;

DATA abt_7;
SET abt_6;
FORMAT first_purchase date9.;
RUN;
```

```
/*age*/

PROC SQL;
CREATE TABLE abt_10 AS
SELECT DISTINCT cust_id, round((2015-YOB)) as age
FROM work.final_data
GROUP BY cust_id;
RUN;
```

```
/*Number of years since being a client*/

PROC SQL;
CREATE TABLE abt_12 AS
SELECT DISTINCT cust_id, round(2015 - customer_since) as client_yrs
FROM work.final_data
GROUP BY cust_id;
RUN;
```

```
/*Quantity of products bough per category*/

PROC SQL;
CREATE TABLE qty_cat as
SELECT DISTINCT cust_id, prod_cat, sum(Qty) as qty_cat
FROM work.final_data
GROUP BY  cust_id, prod_cat;
RUN;


PROC SORT DATA=qty_cat;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=qty_cat
OUT=abt_qty_cat
    PREFIX=qty_;
ID prod_cat;
BY cust_id;
RUN;
```

```
/*Quantity of products bought per customer per city*/

PROC SQL;
CREATE TABLE qty_city as
SELECT DISTINCT cust_id, city, sum(Qty) as qty_city
FROM work.final_data
GROUP BY cust_id, city;
RUN;


PROC SORT DATA=qty_city;
BY cust_id;
RUN;

PROC TRANSPOSE DATA=qty_city
OUT=abt_qty_city
    PREFIX=qty_;
ID city;
BY cust_id;
RUN;
```

```
/*Average total amount spent per customer*/
PROC SQL;
CREATE TABLE avg_amt AS
SELECT DISTINCT cust_id, avg(total_amt) AS avg_amt
FROM work.final_data
GROUP BY cust_id;
RUN;
```

```
/*Customer Segmentation*/

PROC SQL;
CREATE TABLE customer_seg AS
    SELECT DISTINCT cust_id, sum(total_amt) as total_amt,
        CASE
            WHEN SUM(total_amt) <= 10000 THEN 'Bronze'
            WHEN SUM(total_amt) > 10000 AND SUM(total_amt) <= 20000 THEN 'Silver'
            WHEN SUM(total_amt) > 20000 AND SUM(total_amt) <= 30000 THEN 'Gold'
            ELSE 'Platinum'
        END as customer_seg
    FROM work.final_data
    GROUP BY cust_id;
RUN;

/*Lets merge everything for the final table (ABT)*/

DATA abt_final;
MERGE original_var abt_10 abt_7 abt_9 abt_12 avg_amt customer_seg abt_qty_cat abt_qty_pay abt_qty_store abt_freq_cat
 abt_freq_pay freq_sal_chan
 abt_mon_cat abt_mon_subcat abt_mon_pay abt_mon_store;
BY cust_id;
RUN;

DATA final2;
SET abt_final;
ARRAY change numeric;
DO OVER change;
IF change=. THEN change=0;
END;
RUN;
```

Anex 3: Python Code (used for the treatment of data inconsistencies)

```
import pandas as pd
import numpy as np
```

```
data = pd.read_excel('after_imputation_dates.xlsx')
```

```
df = data.copy()
df
```

```
df1 = pd.DataFrame(df['Customer_since'].groupby(df['cust_id']).nunique()).copy()
df1
```

```
df2_bigger = df1[df1['Customer_since'] > 1].copy()
df2_bigger
```

```
lst1 = df2_bigger.index.tolist() #index being the cust_id
#these are the indexes that have more than one customer_since value
```

```
df3_bigger = df.set_index('cust_id').loc[lst1,:].reset_index().drop_duplicates(['cust_id'], keep='first').copy()
df3_bigger
# these are the rows sorted by cust_id where their customer_since have more than one value
```

```
df4_bigger = df3_bigger.set_index('cust_id').copy()
df4_bigger
# df4_is a copy of df3 but with cust_id as the index
```

```
df2_equal = df1[df1['Customer_since'] == 1].copy()
df2_equal
#  here we get the customers wiht only one customer_since value
```

```
lst2 = df2_equal.reset_index()['cust_id'].tolist()
lst2
#  all the cust_ids that have only one customer_since value
```

```
df3_equal = df.set_index('cust_id').loc[lst2,:].reset_index().drop_duplicates(['cust_id'], keep='first').copy()
df3_equal
# here are the rows for each cust_id w only one customer_since value
```

```
df4_equal = df3_equal.set_index('cust_id').copy()
df4_equal
# setting the index as the cust_id
```

```
df = df.set_index('cust_id')
df['Customer_since'] = df4_bigger['Customer_since']
df['Customer_since']
# for each cust_id in df that is in df4_bigger, the value of customer_since was changed
# for the customer_since in df4_bigger
```

```
df['Customer_since'].isna().sum()
```

```
df7_na = df[df['Customer_since'].isna()]
```

```
df7_na = df7_na.reset_index('cust_id')
```

```
df7_na = df7_na.drop_duplicates(['cust_id'], keep = 'first')
```

```
df7_na = df7_na.set_index('cust_id')
```

```
df7_na['Customer_since'] = df4_equal['Customer_since']
```

```
df7_na['Customer_since'].isna().sum()
```

```
df.isna().sum()
```

```
df['Customer_since'] = df['Customer_since'].fillna(df7_na['Customer_since'])
```

```
df.isna().sum()
```

```
df = df.reset_index('cust_id')
```

```
df8 = df[df['Customer_since'] > df['tran_date_year']].copy()
# creates a copy of df with only the rows which have customer_since > tran_date_year

df8['Customer_since'] = df8['tran_date_year']
#now making sure that the values match for each row

df8.loc[:,['cust_id','Customer_since', 'tran_date_year']]
# locating the indexes to later replace this values in the final df
```

```
index8 = df8.index
```

```
df.loc[index8,:] = df8
df.loc[index8,['cust_id','Customer_since', 'tran_date_year']].sort_values('cust_id')
```

```
df[df['Customer_since'] > df['tran_date_year']]
```