Report Mega Market

Data Pre-Processing and Visualization

28/11/2023

Dinis Fernandes 20221848

Inês Santos 20221916

Sara Ferrer 20221947

Vidhi Rajanikante 20221982

Yehor Malakhov 20221691

Introduction

To navigate the terrain of customer-centered retail, Mega Market embarked on a strategic initiative and founded a subgroup of data scientists, whose job is to transform the raw transactional data into a format that allows advanced analysis and the possibility of getting valuable insights.
This report documents the journey of the group of data scientists responsible for the Data Preprocessing processes.
This journey starts in SAS Miner where firstly we will get graphical visualizations of the raw data, then we will treat the outliers and the missing values by eliminating and imputing them, respectively. After SAS Miner, we made a short stop in Python to treat some customer-related inconsistencies. Then we go to SAS Studio, to continue the inconsistencies treatment but this time more related to the sales. After all the inconsistencies are treated, we will begin creating the Analytic-Based Table (ABT) and new variables at the same time. When the ABT is finished, we make another short stop in Python only to create one new variable, and finally, to present some visualizations to Mega Market we will use PowerBi.

Body

Data Analysis

We began by analyzing our raw data to get a better and deeper understanding of the Mega Market company.

To do this we checked the descriptive statistics of all variables, with the node DMDB. During this step we discovered that some variables had missing values, in specific, Age, Monthly Income, and Kids, and Reviews, the other variables had 99999 observations. We also reviewed the categorical variables to determine how many levels (different values) could take, however SAS Miner didn't count higher than 26 levels for Nationality and ProductName, a basic python code revealed that Nationality had 31 different levels and ProductName has 2843 different levels.

Still in the node of descriptive statistics (DMDB) we noticed that we would have at least one outlier in the variable Monthly Income, as it had a large difference between the maximum value and the minimum as well as high standard deviation of 1443,18. Similarly ,in the variable Quantity the minimum and the maximum have a large range, with the maximum value far away from the mean, therefore this value is also probably an outlier. We would also like to point out that the maximum value of the variable Age was 299, which corresponds to an invalid value rather than an outlier, as it is nearly impossible for someone to have lived that long. Having accounted for these variables we decided to pay special attention to them during our analysis. To give a better understanding we present the image of the DMDB node:

Interval Variable Summary Statistics

Variable	Label	Missing	N	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
Age	Age	2266	97733	29	299	54.36	15.12	-0.0963	-0.28
Monthly_Income	Monthly Income	2266	97733	900	90000	2506.40	1443.18	15.9047	582.52
Quantity	Quantity	0	99999	1	100	3.01	1.74	18.9099	1050.29
Total_payed	Total_payed	0	99999	1	77	22.92	17.88	0.9815	0.18
Unit_Price	Unit Price	0	99999	1	15	7.63	4.29	0.1558	-1.19

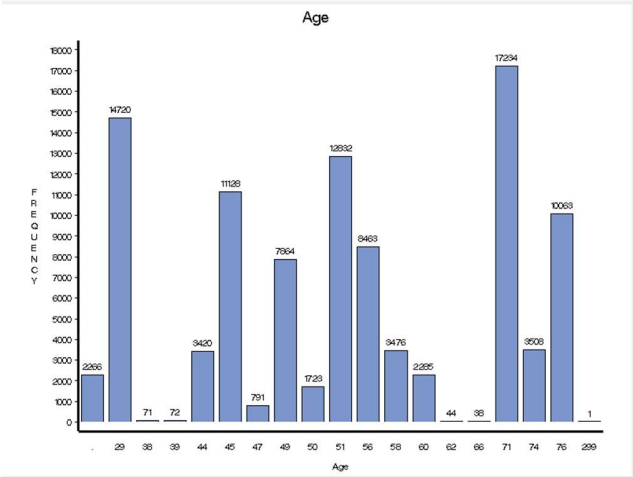
Class Variable Summary Statistics

Variable	Label	Type	Number of Levels	Missing
Channel	Channel	C	2	0
Gender	Gender	C	3	0
Kids	Kids	N	2	2266
Nationality	Nationality	C	26	0
Payment	Payment	C	3	0
ProductName	ProductName	C	26	0
Product_Category_Name	Product Category Name	C	9	0
Reviews	Reviews	N	2	57701

After checking the descriptive statistics of our data, we decided to move on to analyzing the correlation between variables with the node Variable Clustering. This step is highly important to detect multicollinearity and assess the contribution of each variable to our dataset. We also made sure to check for outliers and other data errors.

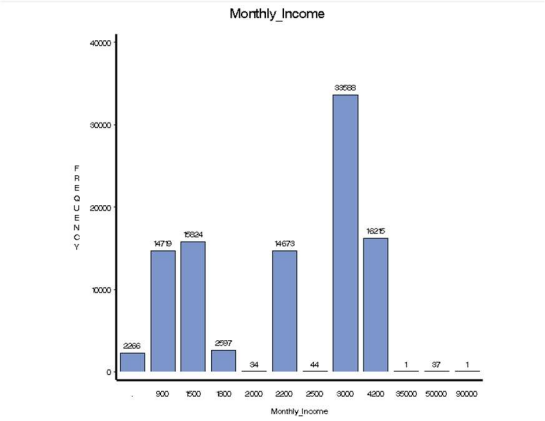
After analyzing the heatmap we concluded that the variables Total_payed and Unit Price are highly correlated, with a correlation of 72% also Total_payed has a correlation of 48% with Quantity. These correlations become clear as the total amount that a customer pay is dependent on the quantity and the price per unit.

After we decided to plot the distribution of the variables with the Multiplot node. We observed that the variable Age doesn't have a defined distribution. Due to this, we will not be able to benefit from a boxplot to analyze outliers.

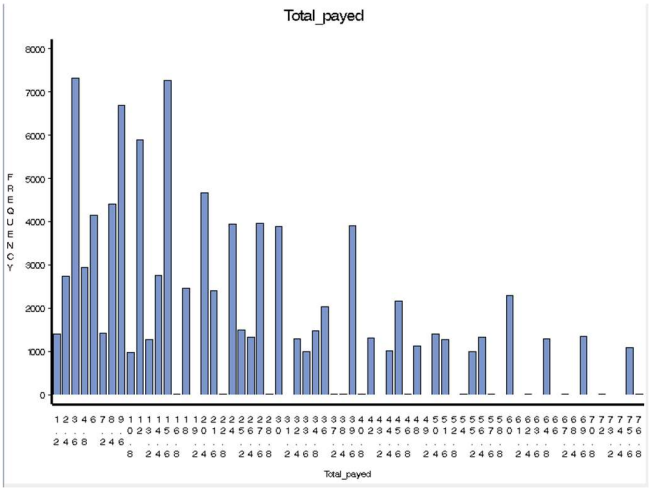


We also analyzed the variable Channel that corresponds to the store type where that transaction took place. We discovered that the transactions in-store made have a lot more representation than the online transactions.

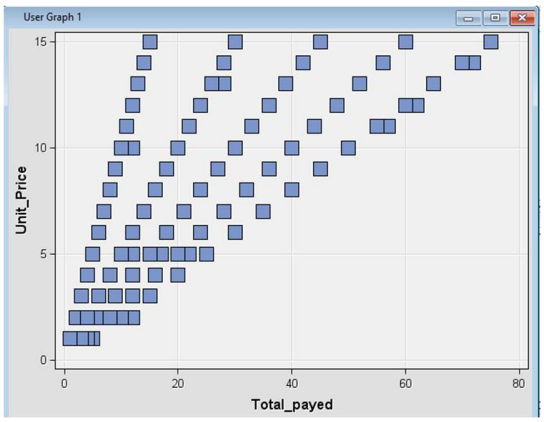
Regarding the Gender variable, we also found out that male customers made a lot more transactions. As for the Kids variable, the plot showed that we had 2266 missing values, which we already knew on the DMDB node, and most of our customers have kids. The variable Monthly Income also doesn't appear to have an evident distribution, as the values vary a lot. However, we can assume that we are in the presence of outliers, possibly the last 3 bins, which is why the variable distribution is not more evident.



Furthermore, we also concluded that most of the transactions were made in the United Kingdom. and the preferable payment type was Paypal, in contrast the least used method was cash. In product categories the most bought was miscellaneous while the less was sombrero. Additionally, the variable Quantity has outliers, with the value 100, specifically 11 transactions with that value. We decided not to analyze the variable reviews since it has a great number of missing values, therefore it's not representative. The next variable, Total Payed, is relatively right skewed. While the Unit Price variable has an approximate uniform distribution, with a peak on 4 and a minimum on 11.



As mentioned before, we will further analyze the correlation between Total_payed, Quantity and Unit-Price. We used the node Graph Explore to create a scatter plot, between Total_Payed and Unit_Price, we observed a positive correlation, which comes to logical sense that the total amount paid in a transaction increase as the unit price increases. However, the correlation is not a positive perfect because if someone purchases a lot of items at a low price will also pay a lot, and someone who buys only one item at a high price will also pay a lot. We could take the same conclusions if we plotted the Total_Payed with Quantity.

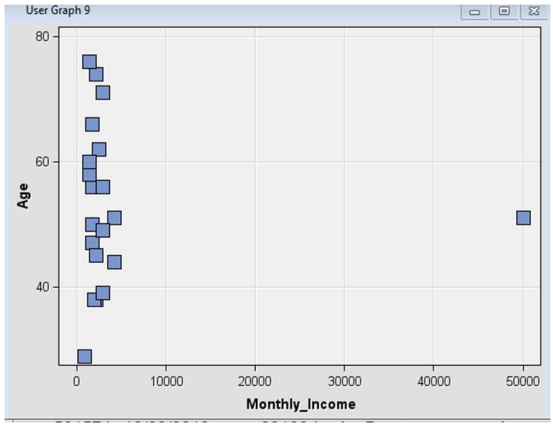


In this node we were not able to use the Boxplot graphs to detect outliers because none of our quantitative variables that we suspected had a normal distribution. Instead of using the boxplot we decided to create scatter plots between the variables we suspected had outliers.

However, the Graph Explore node only represents a sample of the data, nor all outliers or observations are present. We know that the Monthly Income as one person with 90,000, and we also have someone with Age 299

As we can see the graph got squeezed because of the outlier in Montly_Income, also we would like to point out that the graphs made by the node Graph Explore, just use a sample and not all data, therefore that's why we don't have present all outliers, or observations.

As it's possible to view in the previous graph, we know by a fact that monthly income has one person with 90,000, and someone with age 299, but this data does not appear on the graph due to the sampling. Also, we would like to point out that we have an outlier regarding monthly income, however considering the following graph we can say for sure that is not a multidimensional outlier with Age.

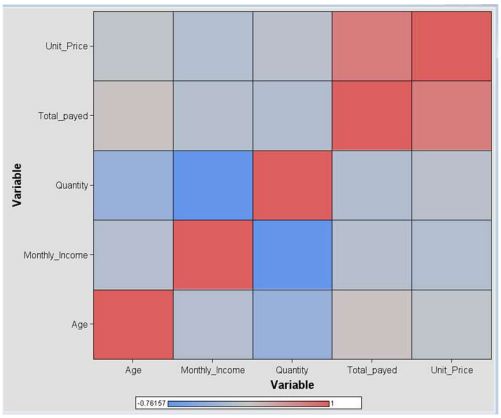


After analyzing our data, we decided to use the node StatExplore to determine the two most bought items. Since the variable has several categories, this is the only analysis that we are going to have, none of the graphs make sense, and even the node multiplot gives an error if we don't drop Product Name. The item most purchased item is Alarm Clock Bakelike Red followed by Alarm Clock Bakelike Ivory. We then decided to deal with our outliers previously discovered.

Outlier Treatment

To treat our outliers, we decided to delete them, always considering the thumb rule of not deleting more than 3% of our data. We used the filter node and then we specified the limits for our variables. We defined Age with a limit of 0 and a maximum of 100, Monthly Income with a minimum of 200 and a maximum of 7000 and Quantity with a minimum of 0 and a maximum of 10. With these limits we only excluded 50 observations, which corresponds to 0.05% of our data which is well below the 3% threshold.

We then decided to analyze the data that only contained outliers, by using the filter node, and changing a parameter of the train, to excluded. We would like to point out that all the 50 transactions/customers were from the United Kingdom and transactions from the category miscellaneous. The correlation between variables, only with outliers, changed slightly, with a high negative correlation between Monthly Income and Quantity. This correlation is a bit strange, as if someone that earns a lot buys very little quantity, this can be due to error values rather than outliers, therefore we decided to further investigate this outlier dataset.



To investigate our outliers, we decided to once again use the multiplot node, to check the distribution, and values of our data. As previously stated, Age doesn't have outliers, but has an error. The Channel still had the same distribution as before, which means Store has more transactions than online. On Gender we conclude that there were no female outliers, or error values. On kids the distribution is also the same as before, we have more people with kids. On Monthly Income we delete the observations that were outside our threshold but we also 11 people that had a monthly income of 900. On payment type the distribution was also the same, the credit card is still the preferable payment type.

After we analyzed our outliers and we have clean data, we decided to impute the missing values.

Missing Values Treatment

To impute our missing values, we decided to use the node Impute, in this Node we used a decision tree as default imputer. The variables that were imputed were Age, Monthly Income and Kids. We didn't apply this transformation before because outliers would have prejudiced our imputing values. Also as stated before, the variable reviews was dropped from our analysis and imputation because it had 79% missing values, with this percentage, as more than half we wouldn't have enough information to predict\impute a good value, it would not be representative of the reality.

Transforming Variables

Next, we decided to apply transformations to some variables to give them a more normal distribution. In Unit Price we opted to apply optimal binning, we also applied Log, into Total Payed as it was left skewed. We didn't export data after these transformations because we would lose interpretability of our Total Payed as the values would change. We would also like to point out that we didn't apply any other transformation to any other variable because our variables weren't skewed.

Final graphical analysis

Now that we have treated the outliers, missing values and applied transformations to the variables that need it, we decided to repeat the process of visualizations that we first did with the raw data, which means using the nodes DMDB, Variable Clustering, Graph Explore, MultiPlot and StatExplore, again.
With the DMDB node we checked again the descriptive statistics of our variables and now after all the variable treatment we can see that we no longer have any missing values nor outliers. Regarding our interval variables, it is noticeable that most are slightly skewed to the left, due to their skewness value being less than zero, which means there is a slightly tendency towards younger ages (IMP_Age), lower number of quantities bought (IMP_Quantity) and a lower amount of money spent in total (LOG_IMP_Total_payed).

Interval Variable Summary Statistics

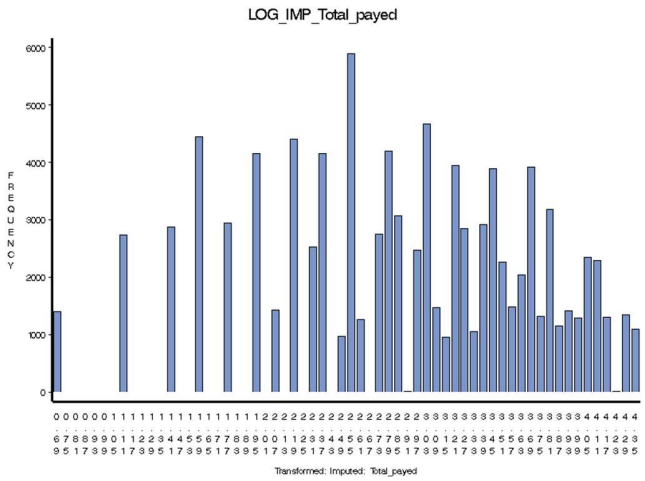
Variable	Label	Missing	N	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
IMP_Age	Imputed: Age	0	99949	29.000	76.00	54.62	15.02	-0.17959	-0.96213
IMP_Monthly_Income	Imputed: Monthly Income	0	99949	900.000	4200.00	2520.68	1080.46	0.06551	-1.05620
IMP_Quantity	Imputed: Quantity	0	99949	1.000	5.00	3.00	1.42	-0.00433	-1.30550
IMP_Unit_Price	Imputed: Unit Price	0	99949	1.000	15.00	7.63	4.29	0.15561	-1.19397
LOG_IMP_Total_payed	Transformed: Imputed: Total_payed	0	99949	0.693	4.36	2.86	0.85	-0.38155	-0.55732

Class Variable Summary Statistics

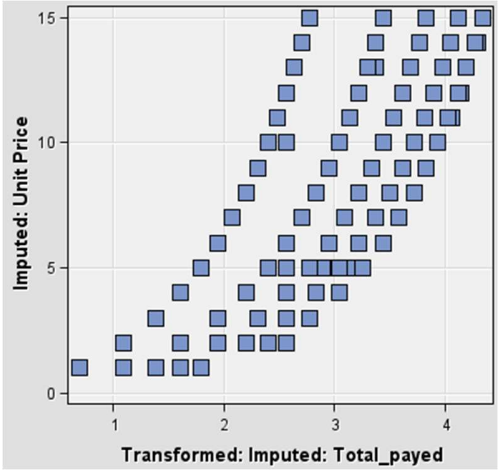
Variable	Label	Type	Number of Levels	Missing
IMP_Channel	Imputed: Channel	C	2	0
IMP_Gender	Imputed: Gender	C	3	0
IMP_Kids	Imputed: Kids	N	2	0
IMP_Nationality	Imputed: Nationality	C	26	0
IMP_Payment	Imputed: Payment	C	3	0
IMP_Product_Category_Name	Imputed: Product Category Name	C	9	0
ProductName	ProductName	C	26	0

Then with the Variable Clustering node, we decide to check again the correlation between variables. To do it, we used a heatmap which was very similar to the one that we presented in the beginning, not having significant changes.

After, we decided to check the graphs of the transformed variables with MultiPlot. Regarding the variable 'LOG_IMP_Total_payed', we can see that its distribution has changed, now it's more similar to a normal distribution. Although this seems better, we will not use this variable with the log transformation as it was mentioned before. Regarding the variable 'IMP_Unit_price', its graph did not change with the transformation applied.

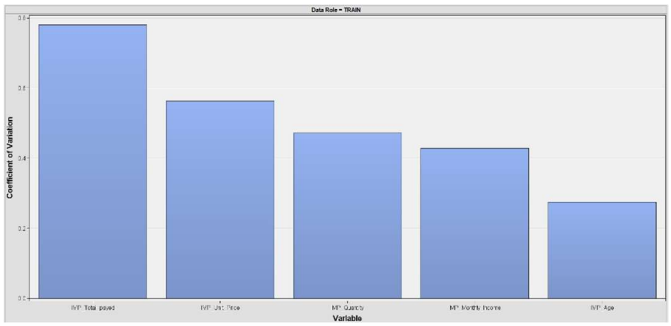


With the Graph Explore node, we decide to again plot a scatter plot between our transformed variables to check if something had changed. We can see that now we have an exponential relationship, which is not interpretable.



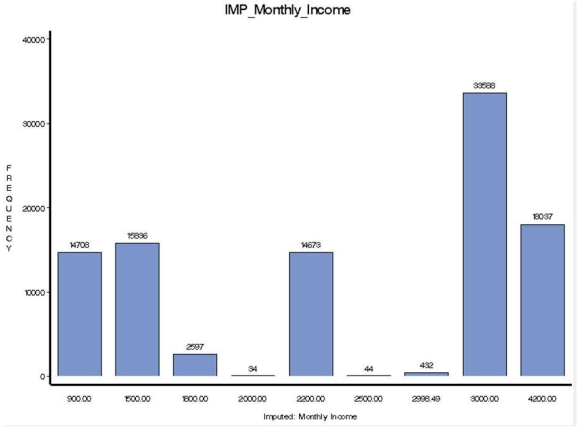
Lastly on the analysis of the transformed variables, we used StatExplore to check if the variable that has the higher percentage in explaining our dataset variability was still the same, and it is.

As we have mentioned before, we did not export the dataset with the transformed variables, so we decided to do again graph analysis to the dataset that we exported. For this analysis we only used the nodes StatExplore and MultiPlot.
With the StatExplore node, we noticed that the Percent of Variability did not change, being the variable 'IMP_Channel' the one that explains our dataset variability more. We also checked that the interval variable with the highest coefficient of variation was 'IMP_Total_payed', this means that it's the variable where the datapoints are more spread out in relation to the mean.



With the MultiPlot node, we can start with the Age graph. In our new graph we have people with the age that is not an integer this is due to the fact that that in imputation there was no rounding made, therefore we cannot analyze the graphs made by the variables that were imputed. Later, on Python we rounded these results.
We also concluded that the categorical variables have the same distribution as before, as they didn't have any outliers and missing values.
Our only binary variable, Kids, although it had missing values, the distribution, as expected, maintained the same.

The variable that is worth noting is Monthly Income, as we treated outliers and imputed missing values, we can now notice that most of our customers earn 3000 per month. We would also like to point out that the difference between the highest income and lower income is 3300.



Checking Coherence

To check the incoherences in our dataset, we used two programming languages, Python and SQL, being the first one used in the jupyter notebooks and the second one in SAS Studio. Firstly, we started with Python to check customer related incoherences:

- If a customer had different genders in different transactions.

```
inconsistent_customers = data.groupby('CustomerNo')['Gender'].nunique() > 1

inconsistent_customers_list = inconsistent_customers[inconsistent_customers].index.tolist()

print(inconsistent_customers_list)

data = data[~data['CustomerNo'].isin(inconsistent_customers_list)]
```

```
[12414, 13841, 16584]
```

We started by grouping the 'Gender' column by 'CustomerNo' and checking if the number of unique values is bigger than 1. If we print 'inconsistent_customers' we get a Pandas Series with all customer numbers and with the value True, if the number of unique values is bigger than 1 and False if it is smaller or equal to 1.

Then we create a list with the customer number of the customers that are in 'inconsistent_customers', which means, the customers where the count of unique gender values are bigger than 1. Then we print this new list, and we can see that the customers 12414, 13841 and 16584 have an inconsistency in the 'Gender' column.

16584	United Kingdom	F	Paypal	Store
16584	United Kingdom	M	Paypal	Store

13841	United Kingdom	M	Credit Card	Store
13841	United Kingdom	F	Credit Card	Store

12414	United Kingdom	M	Paypal	Store
12414	United Kingdom	F	Paypal	Store

Finally, we update our data by only keeping the customers that do not belong to the 'inconsistent_customers_list'.

- In a transaction it was stated that a certain customer had kids and in another the same customer did not have kids.

```
inconsistent_customers_kids = data.groupby('CustomerNo')['IMP_Kids'].nunique() > 1

inconsistent_customers_list_kids = inconsistent_customers_kids[inconsistent_customers_kids].index.tolist()

print(inconsistent_customers_list_kids)
```

```
[]
```

The code is doing the same as the previous, but this time we are grouping the 'IMP_Kids' column by 'CustomerNo'.

When we print the 'inconsistent_customers_list_kids', we get an empty list, which means that we do not have customers with this type of inconsistency.

- If a customer had a different nationality in different transactions.

```
inconsistent_customers_nat = data.groupby('CustomerNo')['Nationality'].nunique() > 1

inconsistent_customers_list_nat = inconsistent_customers_nat[inconsistent_customers_nat].index.tolist()

print(inconsistent_customers_list_nat)
```

```
[]
```

The code is doing the same as the previous ones, but this time we are grouping the 'Nationality' column by 'CustomerNo'.

When we print the 'inconsistent_customers_list_nat', we get an empty list, which means that we do not have customers with this type of inconsistency.

- If a transaction had 'Channel' as Online and 'Payment' as Cash, because if a customer is buying through the online store, it is not possible to pay in cash.

```
print(data["Payment"].unique())
```

```
['Cash' 'Credit Card' 'Paypal']
```

Firstly, we started by checking what different values we had for 'Payment.

```
display(data[(data['Channel'] == 'Online') & (data['Payment'] == 'Cash')])
```

Quantity	Total_payed	CustomerNo	Nationality	Gender	Payment	Channel	Product_Category_ID
3	21	17680	United Kingdom	F	Cash	Online	4
3	9	17680	United Kingdom	F	Cash	Online	2
3	36	17680	United Kingdom	F	Cash	Online	2
4	16	17680	United Kingdom	F	Cash	Online	2
4	52	17680	United Kingdom	F	Cash	Online	2

After, we stated the condition of 'Channel' being 'Online' and 'Payment' being 'Cash', to check which transactions had this type of inconsistency.

```
condition = (data['Channel']=='Online') & (data['Payment']=='Cash')
```

```
data.drop(data[condition].index, inplace=True)
```

```
display(data[(data['Channel'] == 'Online') & (data['Payment'] == 'Cash')])
```

TransactionNo	Date	ProductID	ProductName	Quantity	Total_payed	CustomerNo	Nationality	Gender	Payment	Channel

Then we created a variable with the previous condition, so that we could delete these observations from our data, which is what we are doing in the second line of code.

Lastly, we display once again our condition just to make sure that every observation that had this inconsistency was deleted.

As we have mentioned, we had an issue with a value for the variable Age, so we solved that issue in Python by changing its data type from a float to an integer.

In SAS Studio, we checked the sales related incoherences:

- If 'Unit Price' is lower than 0:

```
4  if (unit_price<0) then do;
5  delete;
6  end;
```

We are just checking if 'Unit price' is lower than 0, if so, we delete that observation because a price cannot be a negative value.

- If 'Total Payed' is lower than 0:

```
8  if(total_payed<0) then do;
9  delete;
10 end;
```

Again, we are just checking if 'Total Payed' is lower than 0 and if so, we delete the observation because the amount of money that a customer pays cannot be negative.

- If 'Quantity' is lower than 0:

```
12 if (quantity<0) then do;
13 delete;
14 end;
```

Again, we are just checking if 'Quantity' is lower than 0, if so, we delete the observation because a quantity can never be a negative value.

- If 'Total Payed' is less/more than the condition 'Unit Price' * 'Quantity':

1)
```
16 if (total_payed < unit_price*quantity) then do;
17 delete;
18 end;
```

2)
```
19
20 if (total_payed > unit_price*quantity) then do;
21 delete;
22 end;
```

We are checking if the value resulting from the multiplication of 'Unit Price' with 'Quantity' is lower (1)/higher (2) than the value of 'Total Payed', if it is, we delete the observation.

Creating Variables and ABT

To create new variables, firstly we used SAS Studio and after the ABT was done we used Python to create one more variable. The ABT was only done in SAS Studio.

The new variables that were created in SAS Studio are:

- Frequency, the count of transactions of each product category per customer:

```
PROC SQL;
CREATE TABLE abt_1 as
SELECT customerno, product_category_name, count(DISTINCT(transactionno)) as frequency
FROM work.no_inc
GROUP BY customerno, product_category_name;
RUN;
```

- Monetary, the total money spent on transactions for each product category per customer.

```
PROC SQL;
CREATE TABLE abt_3 as
SELECT customerno, product_category_name, sum(total_payed) as monetary
FROM work.noinc
GROUP BY customerno, product_category_name;
RUN;
```

- First Purchase, the date of the first purchase per customer.

```
PROC SQL;
CREATE TABLE abt_6 AS
SELECT DISTINCT customerno, min(date) as first_purchase
FROM work.noinc
GROUP BY customerno;
RUN;
```

As this variable did not need to be transposed, we also got some other variables that we already had in the transactional table to our ABT.

```
round(imp_age) as Age, sum(quantity) as Total_Quantity, sum(total_payed) as Total_Spent, nationality as Nationality,

gender as Gender, round(imp_kids) as Kids, round(imp_monthly_income) as Monthly_Income
```

- Last Purchase, the date of the last purchase per customer.

```
PROC SQL;
CREATE TABLE abt_8 AS
SELECT DISTINCT customerno, max(date) as last_purchase
FROM work.no_inc
GROUP BY customerno;
RUN;
```

- Lowest Purchase, the lowest amount spent on a product category per customer.

```
PROC SQL;
CREATE TABLE abt_10 as
SELECT customerno, product_category_name, min(total_payed) as lowest_purchase
FROM work.no_inc
GROUP BY customerno, product_category_name;
RUN;
```

- Highest Purchase, the highest amount spent on a product category per customer.

```
97  PROC SQL;
98  CREATE TABLE abt_12 as
99  SELECT customerno, product_category_name, max(total_payed) as highest_purchase
100 FROM work.no_inc
101 GROUP BY customerno, product_category_name;
102 RUN;
```

- Average Purchase, the average amount spent on a product category per customer.

```
117 PROC SQL;
118 CREATE TABLE abt_14 as
119 SELECT customerno, product_category_name, avg(total_payed) as average_purchase
120 FROM work.no_inc
121 GROUP BY customerno, product_category_name;
122 RUN;
```

- Percentage, the percentage of transactions in each product category for each customer.

```
154 PROC SQL;
155 CREATE TABLE abt_18 as
156 Select customerno, count(transactionno) as total_transactions
157 From work.no_inc
158 Group By customerno;
159 Run;
160
161 PROC SQL;
162 CREATE TABLE abt_19 as
163 Select freq.customerno, product_category_name , round(sum(frequency)/sum(total_transactions)*100) as percentage
164 From work.abt_1 freq inner join work.abt_18 total_t on freq.customerno=total_t.customerno
165 Group By freq.customerno, product_category_name;
166 Run;
```

- Payment Type, if 1 it means that the customer used that type of payment, if 0 the customer did not use that payment type.

```
181 PROC SQL;
182 create table abt_21 as
183 select customerno, payment, count(distinct(transactionno)) as payment_type
184 from work.no_inc
185 group by customerno, payment;
186 run;
```

- Store Type, if 1 it means that the customer bought in that type of store, if 0 the customer did not buy in that store type.

```
201 PROC SQL;
202 create table abt_23 as
203 select customerno, channel, count(distinct(transactionno)) as store_type
204 from work.no_inc
205 group by customerno, channel;
206 run;
```

The ABT was being, made while we created the new variables, because every time we created a new variable, we would create a new table, then sort it by customer and then transpose it so we could have by customer number instead of transaction number.

```
PROC SORT DATA=abt_1;
    BY customerno;
RUN;

PROC TRANSPOSE DATA=abt_1
    OUT=abt_2
    PREFIX=freq_;
    ID product_category_name;
    BY customerno;
RUN;
```

After the creation of all the new variables, we merged them all into a new table so that after we could change the values that were None to 0.

```
DATA final_abt;
    MERGE abt_5 abt_7 abt_9 abt_17 abt_20 abt_22 abt_24;
    BY customerno;
    Drop _name_;
RUN;
```

```
DATA final;
    SET final_abt;
    ARRAY change _numeric_;
        DO OVER change;
            IF change= . THEN change = 0;
        END;
RUN;
```

These None values are, for example, when a customer never bought any item of a certain category then when we calculate the frequency for that category the result will be None.

freq_Office supplies
.
1
.
1
.
.

After the ABT was done, we exported it and created the last new variable in a Jupyter Notebook. This new variable is called ‘Category’ and it gives us the membership status of each customer. If the total money spent of a customer is less than 344€ is classified as Bronze if it is between 344€ and 895€ is classified as Silver and if a customer spent in total more than 895€ is classified as Gold.

The code that we did to create this variable is the follow:

```
bronze_threshold = data['Total_Spent'].quantile(0.33)
silver_threshold = data['Total_Spent'].quantile(0.66)
```

We started by establishing the threshold for the Bronze and Silver categories based on the 33rd percentile (quantile 0.33) and 66th percentile (quantile 0.66) of the ‘Total Spent’ column, respectively.

```
data['Category'] = pd.cut(
    data['Total_Spent'],
    bins=[-float('inf'), bronze_threshold, silver_threshold, float('inf')],
    labels=['Bronze', 'Silver', 'Gold']
)
```

Then, to create the variable ‘Category’ we are creating bins to segment the values of ‘Total Spent’ according to the thresholds we set before and lastly, we are giving labels to the bins (Bronze, Silver, Gold).

PowerBI Analysis

To better visualize our data after going through all the pre-processing processes, we used PowerBI. To do the data analysis we used the table that we got after checking all the incoherences, which means we are not using the ABT. We decided to not use the ABT because almost all the variables are related to one product category and that does not give us that many insights.

As we decided to use the table before the ABT we did not have the variables that we created so we had to create some of them again and some new ones but this time as we are working with PowerBi we made them as measures:

- **avg_age_round**, this measure is calculating the average age of the customers and rounding its value.

```
avg_age_round = ROUND(AVERAGE(NOINC2[IMP_Age]),0)
```

- **avg_quantity_per_transaction**, in this measure we are calculating the average quantity bought per transaction and rounding its value.

```
avg_quantity_per_transaction = ROUND(SUM(NOINC2[Quantity])/DISTINCTCOUNT(NOINC2[TransactionNo]),0)
```

- **avg_spent_cust**, this measure is calculating the average money spent per customer.

```
avg_spent_cust = SUM(NOINC2[Total_payed])/DISTINCTCOUNT(NOINC2[CustomerNo])
```

- **max_spent**, in this measure we get the maximum amount spent in a transaction.

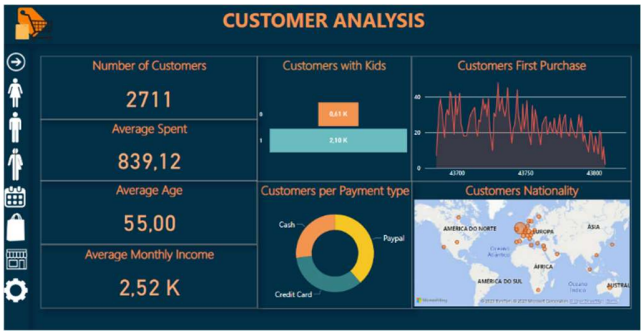
```
max_spent = MAX(NOINC2[Total_payed])
```

- **first_purchase_date**, for this one we created a column instead of a measure because we wanted to get a value for each row. This column gives us the date of the first purchase of each customer.

```
first_purchase_date = CALCULATE(MIN(NOINC2[Date]),ALLEXCEPT(NOINC2, NOINC2[CustomerNo]))
```

We made 3 different pages, one for customer analysis, another for sales analysis and the last one for product analysis. In all of 3 pages, we have buttons that allows the company to see our analysis regarding only the gender of the customers (female, male, other), with the calendar button we can see the analysis in a specified period of time, with the shop button we can filter by the product category name, with the store button we can filter by store type and the engine button is only in the customers page, because it only serves the purpose of changing the 'Customers per Payment type' to 'Customers per Store type'.

Customer Analysis:

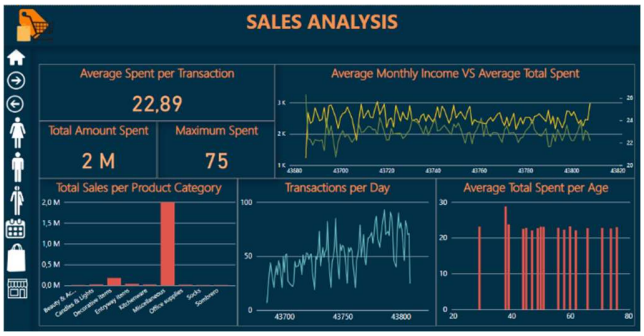


The funnel chart, 'Customers with Kids', shows that most of our customers have kids. We have 2 donut charts, 'Customers per Payment type' and 'Customers per Store type'. The first one shows us that there is not that big of a difference regarding the payment method but the second one tells us that we have more customers buying in physical stores instead of online stores.

The area graph, 'Customers First Purchase', shows us the days where first purchases happened and the number of customers who made their first purchase on a day. We can see that we have a high at the end of September, the exact day is the twenty-second with 48 customers making their first purchase.

Lastly, we have a map that shows us where our customers are from and we can conclude that most are from Europe, more precisely the United Kingdom.

Sales Analysis:



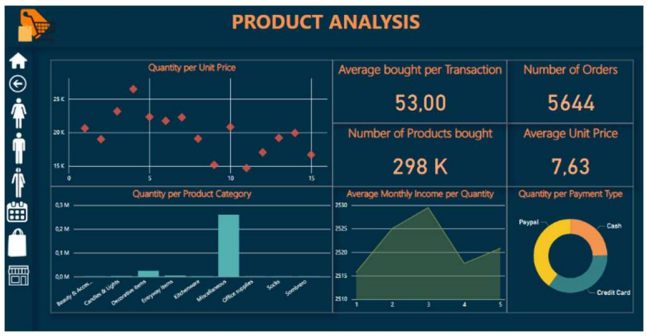
In the stacked line chart, 'Average Monthly Income VS Average Total Spent', the green line shows us the average of the total amount spent in the products and the yellow line shows us the customers average monthly income. We made this graph because we wanted to check if there were cases where the total spent was higher than the monthly income which appears to happen when the green line is a little higher than the yellow one, but it does not because the lines are in different scales. We wanted to check these cases because they could be seen as inconsistencies but also not because a customer may spend more than what it receives monthly if he saves money from his last monthly income.

The line chart, 'Transactions per Day', shows us the count of transactions per day. We can see that we have some highs in the end of September/beginning of October and also from the beginning of November until the beginning of December probably because the customers started buying Christmas presents.

The ‘Total Sales per Product Category’ bar chart shows us the total money spent in each product category. Our top category is Miscellaneous with a total amount of almost 2 million and the second best is Decorative items with almost 0,2 Million.

The bar chart, ‘Average Total Spent per Age’, shows us how much our customers spent on average regarding their age. We can see that our high is with customers of the age of around forty, more precisely 38 years.

Product Analysis:



The scatter plot, ‘Quantity per Unit Price’, shows us how the quantity of products bought vary with their price per unit. We have a high at unit price 4 with a total quantity bought of 26462 products and the lowest is at unit price 11 with 14708 products.

The ‘Quantity per Product Category’ bar chart shows us the total quantity bought of each product category. Our top category is Miscellaneous, and the second best is Decorative items.

The area graph, ‘Average Monthly Income per Quantity’, shows the relation between the average of our customers monthly income and the quantity of product bought. Our highest is the quantity of products bought equal to 3 and the average monthly income is 2529,50.

Lastly, we have a donut chart, ‘Quantity per Payment Type’, that shows us the quantity of products bought regarding the payment type used. We can state that the quantity of products bought is bigger with the payment type Paypal.

Conclusion

In the exploration of Mega Market's company, we aimed to analyze, transform, and create an ABT to improve decision making and to provide significant insights of the company.
Firstly, we analyzed the initial dataset provided, where we concluded that some variables would not be useful for our analysis, such as ProductName and Reviews. In our initial analysis we also quickly found out that we had missing values, outliers and even irregularities in our dataset, all these problems were rapidly solved later in our project. Still in the SAS Enterprise Miner software we filtered outliers and even analyzed them, so we could make a characterization of our best customers, we then imputed missing values with decision trees and finally applied transformations to our data to make it more like a normal distribution.
We then moved to Python, where we checked incoherences, and then in SAS Studio, in total we deleted 434 observations that were not consistent, therefore increasing the credibility and value of our data. Still in SAS studio we created an Analytical Base Table to help solve business problems but also to help improve the accuracy of machine learning models. After creating the ABT we went again to python to segment customers in Bronze, Silver and Gold. This segmentation was highly important to provide a better understanding of our customers' loyalty and affinity for Mega Market.
Finally, we joined everything on PowerBI facilitating interactivity with the dashboards, which provide easy-to-understand insights of the company.
As a result, Mega Market now has a deeper understanding, with powerful insights of the characterization of our most loyal customers, but also about our sales, trends, and the most bought products, having also available an ABT that can be used for future projects improving the results but also refining the decision-making.