
# Customer Personality Analysis

Team : Fastai

## Overview

Customer Personality Analysis is a detailed analysis of a company's ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers.

Customer personality analysis helps a business to modify its product based on its target customers from different types of customer segments. For example, instead of spending money to market a new product to every customer in the company's database, a company can analyze which customer segment is most likely to buy the product and then market the product only on that particular segment.

## Problem Statement

We have to study the dataset and divide the dataset into segments so that we are able to better understand the behavior of the customers .

The purpose of customer segmentation is to divide customers into many different ways. Customers can be grouped by their demographic, behavior, lifestyle, psychographic, value, etc.

Segmentation is mostly used for marketing, but there are other reasons to segment your customer base. Using customer segmentation in marketing means that you can target the right people with the right messaging about your products . For example, instead of spending money to market a new product to every customer in the company's database, a company can analyze which customer segment is most likely to buy the product and then market the product only on that particular segment .

This will increase the success of marketing campaigns.

## Existing Solutions

The existing methods of customer personality analysis include the following:

- The traditional method of customer personality analysis where customer personalities are classified into 4 types i.e. Driver, Analytical, Expressive and Amiable
- Customer Personality Prediction using the Ensemble Technique: In this approach, the authors created the ensemble model by combining support vector machine, naive bayes, logistic regression, KNN and gradient boost.
- Cluster analysis and customer ranking using K-medoids clustering (PAM algorithm)
- Customer personality analysis k-means and agglomerative clustering

## Our Approach

Our team started by following a step by step process . We decided to divide the problem statement into smaller segments and decided to tackle each segment step by step.

1. Exploratory Data Analysis
2. Data visualization
3. Feature engineering
4. Defining the clusters

These were the steps we decided to follow for the project.

# 1. Exploratory Data Analysis :

## Data :

The dataset for this project was downloaded from the Kaggle **Customer personality analysis .**

Our dataset contains the following customer features:

1. People
2. Products
3. Promotion
4. Place

## Content :

**People**

- ID: Customer's unique identifier
- Year_Birth: Customer's birth year
- Education: Customer's education level
- Marital_Status: Customer's marital status
- Income: Customer's yearly household income
- Kidhome: Number of children in customer's household
- Teenhome: Number of teenagers in customer's household
- Dt_Customer: Date of customer's enrollment with the company
- Recency: Number of days since customer's last purchase
- Complain: 1 if the customer complained in the last 2 years, 0 otherwise

**Products**

- MntWines: Amount spent on wine in last 2 years
- MntFruits: Amount spent on fruits in last 2 years
- MntMeatProducts: Amount spent on meat in last 2 years
- MntFishProducts: Amount spent on fish in last 2 years
- MntSweetProducts: Amount spent on sweets in last 2 years
- MntGoldProds: Amount spent on gold in last 2 years

**Promotion**

- NumDealsPurchases: Number of purchases made with a discount
- AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise
- AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
- AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
- AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise
- AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise
- Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

**Place**

- NumWebPurchases: Number of purchases made through the company's website
- NumCatalogPurchases: Number of purchases made using a catalogue
- NumStorePurchases: Number of purchases made directly in stores
- NumWebVisitsMonth: Number of visits to company's website in the last month

## Loading Libraries :

In [1]:
```
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import sklearn
import seaborn as sns
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from mpl_toolkits.mplot3d import Axes3D
from yellowbrick.cluster import KElbowVisualizer

import warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

## Loading the Datasets :

In [2]:
```
#load our dataset and add delimiter to separate it
customer_data=pd.read_csv("marketing_campaign.csv",sep="\t",low_memory=False)
#View a summary of our dataset
customer_data.head()
```

Out[2]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Recency | MntWines | ... | NumWebVisitsMonth | AcceptedCmp3 | AcceptedCmp4 | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5524 | 1957 | Graduation | Single | 58138.0 | 0 | 0 | 04-09-2012 | 58 | 635 | ... | 7 | 0 | 0 | |
| **1** | 2174 | 1954 | Graduation | Single | 46344.0 | 1 | 1 | 08-03-2014 | 38 | 11 | ... | 5 | 0 | 0 | |
| **2** | 4141 | 1965 | Graduation | Together | 71613.0 | 0 | 0 | 21-08-2013 | 26 | 426 | ... | 4 | 0 | 0 | |
| **3** | 6182 | 1984 | Graduation | Together | 26646.0 | 1 | 0 | 10-02-2014 | 26 | 11 | ... | 6 | 0 | 0 | |
| **4** | 5324 | 1981 | PhD | Married | 58293.0 | 1 | 0 | 19-01-2014 | 94 | 173 | ... | 5 | 0 | 0 | |

5 rows × 29 columns

Data Information :

In [4]:
```python
#Check information about our data
customer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

Checking for missing values :

### Check for Missing Values

In [5]:
```
#Check for missing values in %
customer_data.isna().sum() / len(customer_data) * 100
```

Out[5]:
```
ID                     0.000000
Year_Birth             0.000000
Education              0.000000
Marital_Status         0.000000
Income                 1.071429
Kidhome                0.000000
Teenhome               0.000000
Dt_Customer            0.000000
Recency                0.000000
MntWines               0.000000
MntFruits              0.000000
MntMeatProducts        0.000000
MntFishProducts        0.000000
MntSweetProducts       0.000000
MntGoldProds           0.000000
NumDealsPurchases      0.000000
NumWebPurchases        0.000000
NumCatalogPurchases    0.000000
NumStorePurchases      0.000000
NumWebVisitsMonth      0.000000
AcceptedCmp3           0.000000
AcceptedCmp4           0.000000
AcceptedCmp5           0.000000
AcceptedCmp1           0.000000
AcceptedCmp2           0.000000
Complain               0.000000
Z_CostContact          0.000000
Z_Revenue              0.000000
Response               0.000000
dtype: float64
```

**Conclusion :**

- Our dataset comprises 2240 observations and 29 features or can be stated as 2240 rows and 29 columns.
- From our dataset we have missing values only in the income column. Representing 1.1% of all the captured income.
- Customer's enrollment date column has an incorrect data type. It's supposed to be date time.We shall need to convert it.
- We can handle the missing values using the imputation method by replacing them with the median income from the customers captured.

## Data Cleaning :

### Handling Missing Values :

To tackle the missing values in the income column we will replace them with the  median income from the customers in the data .

In [7]:
```
#Handling missing values
customer_data["Income"] = customer_data["Income"].fillna(customer_data['Income'].median())
```

### Changing customer’s enrollment date column to the correct data type :

Customer’s enrollment date column has an incorrect data type it is in form of object we need to change it to a date time format .

In [8]:
```
#Change Customer's enrollment date  to the correct data type which is date
customer_data['Dt_Customer']=pd.to_datetime(customer_data['Dt_Customer'],format='%d-%m-%Y')
```

## More information about dataset :

In [11]:
```
#Customer's Education and Marital Status categories
print('Customers Education Categories: \n',customer_data['Education'].value_counts())
print('\n Customers Marital Status Categories: \n',customer_data['Marital_Status'].value_counts())
```

```
Customers Education Categories: 
 Graduation    1127
PhD            486
Master         370
2n Cycle       203
Basic           54
Name: Education, dtype: int64

 Customers Marital Status Categories: 
 Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

In [12]:
```
#Given the various marital statuses we can work with the following (Single,Married,Divorsed,Widow) as recognised widely.
customer_data['Marital_Status']=customer_data['Marital_Status'].replace({'Together':'Married','Alone':'Single','Alone':'Single','Absurd
#Replace 2n Cycle column with Master
customer_data['Education']=customer_data['Education'].replace({'2n Cycle':'Master'})
#Drop columns
customer_data.drop(['Z_CostContact','Z_Revenue'],axis=1,inplace=True)
```

## Conclusion :

- There are no duplicates in our dataset.
- We can drop the ID column as it has no correlation to any feature in the dataset as it's randomly generated by the system.
- Our dataset comprises integer(25),float(1),object(2) and datetime(1) data types. The majority being integer data type.
- We can remove the following columns(Z_CostContact and Z_Revenue) as we don't have information about them even from our data dictionary.
- Looking at the marital status column we shall need to reduce the categories to the widely known statuses as (Single,Married,Divorced,Widow).
- When we look at the education level types the 2n cycle is similar to Masters hence we need to remove this column.
- There are categorical values that we shall need to encode later after we perform feature engineering and EDA first we look at them.

# Feature Engineering :

- Given the customer's birth year we shall calculate their age as per 2014 when their details were being captured.
- Add a dollar sign to the income column for clarity purposes.
- Given there are customers who are parents to kids and teens we can create a column for family size and also number of children
- The Education level we can create another column named post_graduate to capture those with (Master + PhD) as 1 and 0 for undergraduate + Basic.
- Given there are customers who are not parents despite their marital status not being Single we can create a column for Parent
- Given the various amounts spent on particular product categories by a customer we can create a column for Customer total Purchase.

In [13]:

```
#Calculate Customer Age using 2014 as the current year
customer_data['Customer_Age']=2014-customer_data['Year_Birth']
#Rename Income column add a dollar sign
customer_data.rename(columns={'Income':'Income($)'},inplace=True)
#Create column to handle post graduate details
customer_data['Postgraduate']=customer_data['Education'].replace({'PhD':1,'Master':1,'Graduation':0,'Basic':0})
customer_data['Postgraduate']=customer_data['Postgraduate'].astype(int)
#Calculate customer's number of children
customer_data['no_of_children']=customer_data['Kidhome']+customer_data['Teenhome']
#Obtain infomation if the  customer is a parent(1-is a parent,0 is not a parent)
customer_data['Parent']=customer_data['no_of_children'].replace({1:1,0:0,2:1,3:1})
#Calculate the family Size
customer_data['family_size']=customer_data['Marital_Status'].replace({'Single':1,'Married':2,'Divorced':1,'Widow':1})+customer_data['no
#Calculate the total Customer Purchase based on amount spent on listed products
customer_data['custm_tot_purchase($)']=customer_data['MntWines']+customer_data['MntFruits']+customer_data['MntMeatProducts']+customer_d
customer_data.drop(['Year_Birth'],axis=1,inplace=True)
```

## Showing the dataset :

In [14]:
```
customer_data.describe().T
```

Out[14]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Income($)** | 2240.0 | 52237.975446 | 25037.955891 | 1730.0 | 35538.75 | 51381.5 | 68289.75 | 666666.0 |
| **Kidhome** | 2240.0 | 0.444196 | 0.538398 | 0.0 | 0.00 | 0.0 | 1.00 | 2.0 |
| **Teenhome** | 2240.0 | 0.506250 | 0.544538 | 0.0 | 0.00 | 0.0 | 1.00 | 2.0 |
| **Recency** | 2240.0 | 49.109375 | 28.962453 | 0.0 | 24.00 | 49.0 | 74.00 | 99.0 |
| **MntWines** | 2240.0 | 303.935714 | 336.597393 | 0.0 | 23.75 | 173.5 | 504.25 | 1493.0 |
| **MntFruits** | 2240.0 | 26.302232 | 39.773434 | 0.0 | 1.00 | 8.0 | 33.00 | 199.0 |
| **MntMeatProducts** | 2240.0 | 166.950000 | 225.715373 | 0.0 | 16.00 | 67.0 | 232.00 | 1725.0 |
| **MntFishProducts** | 2240.0 | 37.525446 | 54.628979 | 0.0 | 3.00 | 12.0 | 50.00 | 259.0 |
| **MntSweetProducts** | 2240.0 | 27.062946 | 41.280498 | 0.0 | 1.00 | 8.0 | 33.00 | 263.0 |
| **MntGoldProds** | 2240.0 | 44.021875 | 52.167439 | 0.0 | 9.00 | 24.0 | 56.00 | 362.0 |
| **NumDealsPurchases** | 2240.0 | 2.325000 | 1.932238 | 0.0 | 1.00 | 2.0 | 3.00 | 15.0 |
| **NumWebPurchases** | 2240.0 | 4.084821 | 2.778714 | 0.0 | 2.00 | 4.0 | 6.00 | 27.0 |
| **NumCatalogPurchases** | 2240.0 | 2.662054 | 2.923101 | 0.0 | 0.00 | 2.0 | 4.00 | 28.0 |
| **NumStorePurchases** | 2240.0 | 5.790179 | 3.250958 | 0.0 | 3.00 | 5.0 | 8.00 | 13.0 |
| **NumWebVisitsMonth** | 2240.0 | 5.316518 | 2.426645 | 0.0 | 3.00 | 6.0 | 7.00 | 20.0 |
| **AcceptedCmp3** | 2240.0 | 0.072768 | 0.259813 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp4** | 2240.0 | 0.074554 | 0.262728 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp5** | 2240.0 | 0.072768 | 0.259813 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp1** | 2240.0 | 0.064286 | 0.245316 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp2** | 2240.0 | 0.013393 | 0.114976 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **Complain** | 2240.0 | 0.009375 | 0.096391 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **Response** | 2240.0 | 0.149107 | 0.356274 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **Customer_Age** | 2240.0 | 45.194196 | 11.984069 | 18.0 | 37.00 | 44.0 | 55.00 | 121.0 |
| **Postgraduate** | 2240.0 | 0.472768 | 0.499369 | 0.0 | 0.00 | 0.0 | 1.00 | 1.0 |
| **no_of_children** | 2240.0 | 0.950446 | 0.751803 | 0.0 | 0.00 | 1.0 | 1.00 | 3.0 |
| **Parent** | 2240.0 | 0.715179 | 0.451430 | 0.0 | 0.00 | 1.0 | 1.00 | 1.0 |
| **family_size** | 2240.0 | 2.595089 | 0.906959 | 1.0 | 2.00 | 3.0 | 3.00 | 5.0 |
| **custm_tot_purchase($)** | 2240.0 | 605.798214 | 602.249288 | 5.0 | 68.75 | 396.0 | 1045.50 | 2525.0 |

## Results :

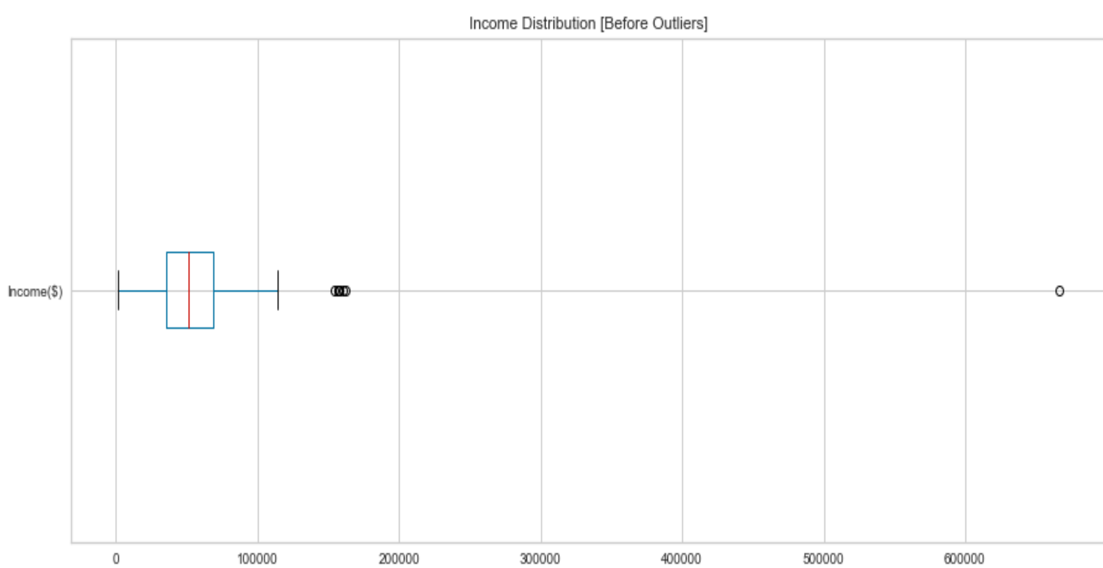
- Worth noting is the large difference between 75th percentile and max values of these columns MntWines,MntFruits, MntMeatProducts,MntSweetProducts,MntGoldProds,Customer_Age.
- This is an indication of outlier values in these columns which we will need to drop them.

## Checking for the outliers :

- These are the extreme values within the dataset. That means the outlier data points vary greatly from the expected values either being much larger or significantly smaller.
- We decided to use boxplots to find out the outliers later we shall drop them at the mark showb by showfliers = False.

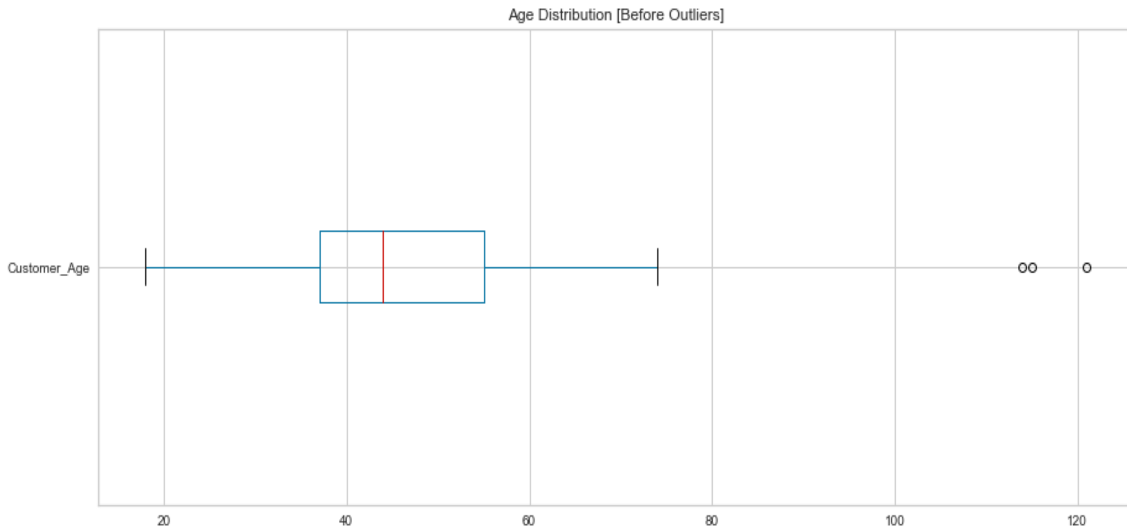
Income distribution :

In [15]:
```
fig, ax = plt.subplots(figsize = [15, 6])
customer_data["Income($)"].plot(kind = "box", vert = False)
plt.title("Income Distribution [Before Outliers]");
```



Age distribution :

In [16]:
```
fig, ax = plt.subplots(figsize = [15, 6])
customer_data["Customer_Age"].plot(kind = "box", vert = False)
plt.title("Age Distribution [Before Outliers]");
```



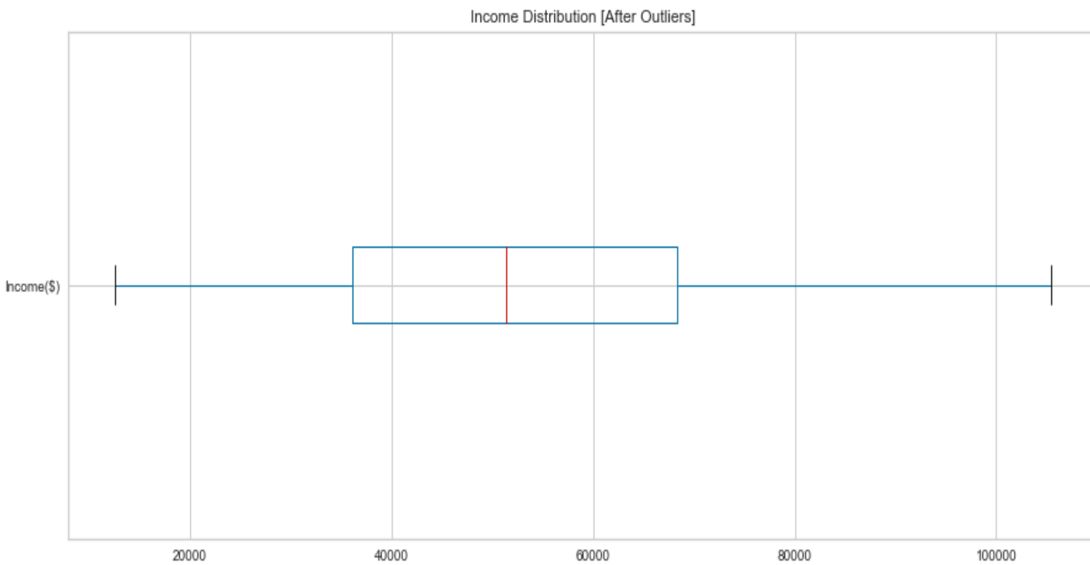

Removing the outliers :

In [17]:
```
#Remove outliers in Age and Income
mask_income = customer_data["Income($)"].between(12500,110000)
mask_age = customer_data["Customer_Age"] <= 75
customer_data = customer_data[mask_income & mask_age]
```

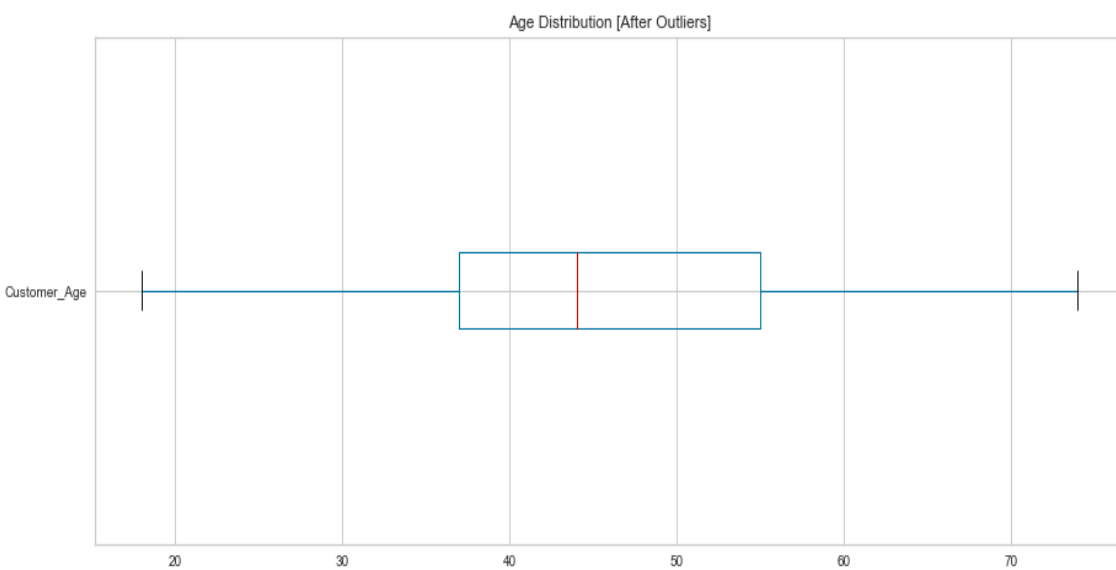
- We removed the outliers and ranged the data between 12,500 and 1,10,000 for the income column .
- Similarly , for the age data we removed all the age data above 75 years .

## After removing the outliers :

In [18]:
```
fig, ax = plt.subplots(figsize = [15, 6])
customer_data["Income($)"].plot(kind = "box", vert = False)
plt.title("Income Distribution [After Outliers]");
```




In [19]:
```
fig, ax = plt.subplots(figsize = [15, 6])
customer_data["Customer_Age"].plot(kind = "box", vert = False)
plt.title("Age Distribution [After Outliers]");
```

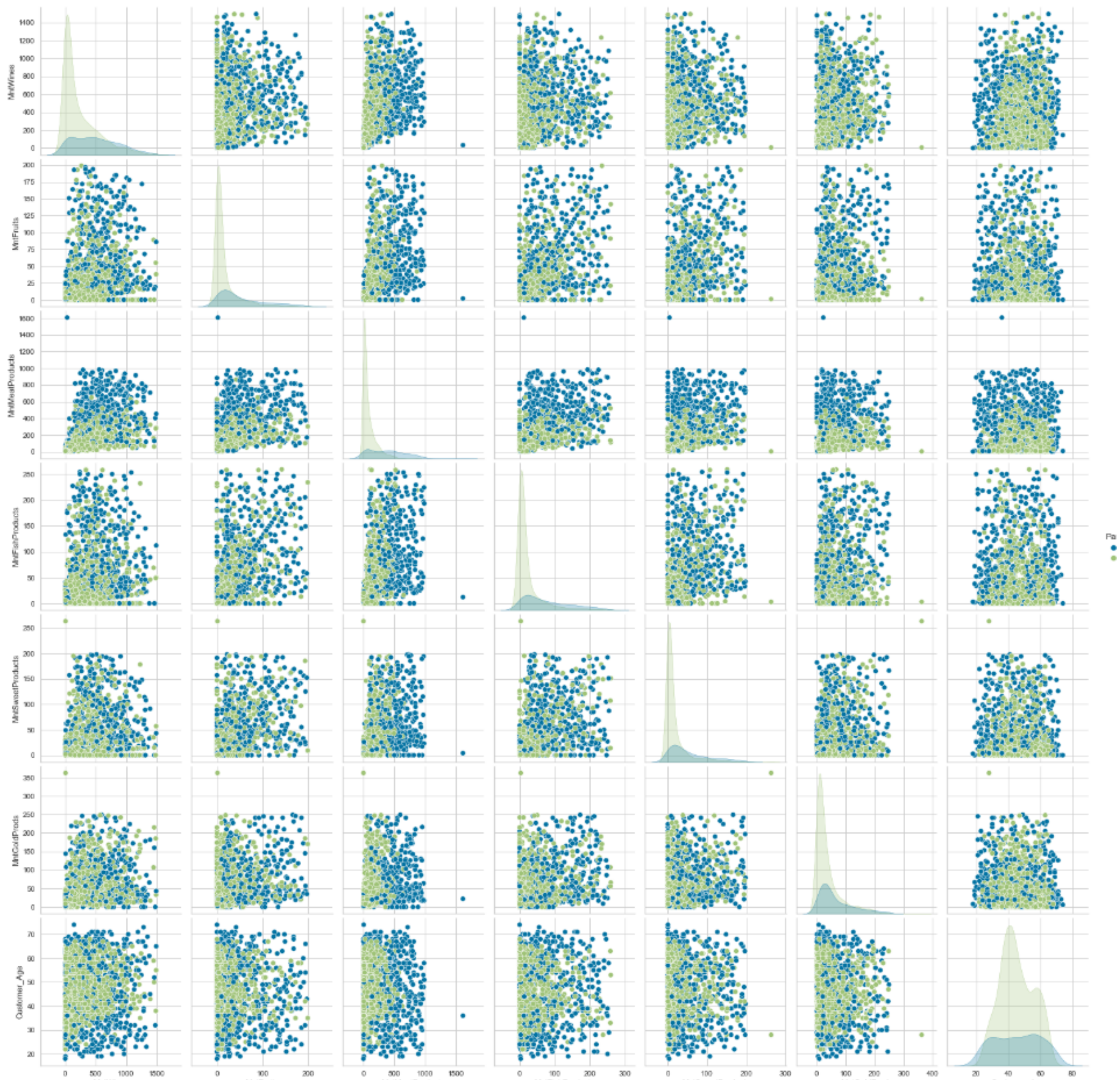


Singling out the columns that have outliers :

In [23]:
```
#Singling out the columns that have the outliers.
customer_purchase=customer_data[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
                                'MntGoldProds','Customer_Age','Parent']]
```

In [24]:
```
#Plotting customer purchases based on the columns that have outliers and using hue as Parent.
sns.pairplot(data=customer_purchase,hue='Parent',height=3)
```

Out[24]: <seaborn.axisgrid.PairGrid at 0x298aab8f250>



After removing the outliers :

In [25]:
```
#Getting the statistical representation following dropping of outliers
customer_data.describe().T
```

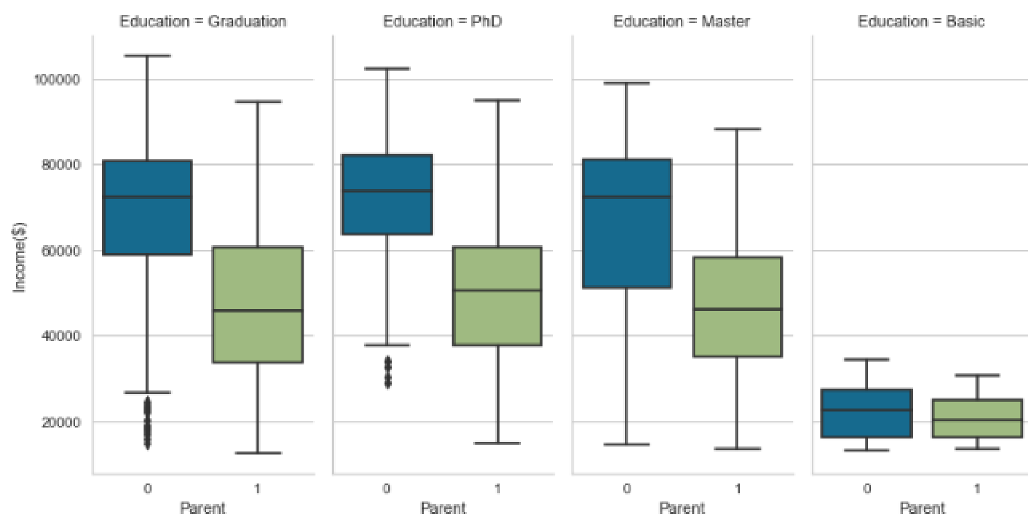
Out[25]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Income($)** | 2192.0 | 52313.816150 | 19935.426539 | 12571.0 | 36058.25 | 51400.5 | 68363.25 | 105471.0 |
| **Kidhome** | 2192.0 | 0.442974 | 0.539144 | 0.0 | 0.00 | 0.0 | 1.00 | 2.0 |
| **Teenhome** | 2192.0 | 0.511405 | 0.544553 | 0.0 | 0.00 | 0.0 | 1.00 | 2.0 |
| **Recency** | 2192.0 | 49.193431 | 28.930345 | 0.0 | 24.00 | 50.0 | 74.00 | 99.0 |
| **MntWines** | 2192.0 | 309.896442 | 337.436448 | 0.0 | 26.00 | 183.5 | 509.00 | 1493.0 |
| **MntFruits** | 2192.0 | 26.684763 | 39.997382 | 0.0 | 2.00 | 8.0 | 34.00 | 199.0 |
| **MntMeatProducts** | 2192.0 | 167.099909 | 217.751892 | 1.0 | 16.00 | 69.0 | 237.25 | 1607.0 |
| **MntFishProducts** | 2192.0 | 38.187044 | 54.995404 | 0.0 | 3.00 | 12.0 | 51.00 | 259.0 |
| **MntSweetProducts** | 2192.0 | 27.417427 | 41.278468 | 0.0 | 1.00 | 9.0 | 35.00 | 263.0 |
| **MntGoldProds** | 2192.0 | 44.237682 | 51.556429 | 0.0 | 9.00 | 25.0 | 57.00 | 362.0 |
| **NumDealsPurchases** | 2192.0 | 2.302007 | 1.822081 | 0.0 | 1.00 | 2.0 | 3.00 | 15.0 |
| **NumWebPurchases** | 2192.0 | 4.109945 | 2.667014 | 0.0 | 2.00 | 4.0 | 6.00 | 27.0 |
| **NumCatalogPurchases** | 2192.0 | 2.660584 | 2.751828 | 0.0 | 0.00 | 2.0 | 4.00 | 11.0 |
| **NumStorePurchases** | 2192.0 | 5.879562 | 3.223332 | 0.0 | 3.00 | 5.0 | 8.00 | 13.0 |
| **NumWebVisitsMonth** | 2192.0 | 5.271898 | 2.280226 | 0.0 | 3.00 | 6.0 | 7.00 | 10.0 |
| **AcceptedCmp3** | 2192.0 | 0.072993 | 0.260184 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp4** | 2192.0 | 0.076186 | 0.265356 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp5** | 2192.0 | 0.073905 | 0.261676 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp1** | 2192.0 | 0.065693 | 0.247802 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **AcceptedCmp2** | 2192.0 | 0.013686 | 0.116211 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **Complain** | 2192.0 | 0.009124 | 0.095105 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **Response** | 2192.0 | 0.151004 | 0.358134 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **Customer_Age** | 2192.0 | 45.234945 | 11.665778 | 18.0 | 37.00 | 44.0 | 55.00 | 74.0 |
| **Postgraduate** | 2192.0 | 0.471715 | 0.499313 | 0.0 | 0.00 | 0.0 | 1.00 | 1.0 |
| **no_of_children** | 2192.0 | 0.954380 | 0.753340 | 0.0 | 0.00 | 1.0 | 1.00 | 3.0 |
| **Parent** | 2192.0 | 0.716697 | 0.450705 | 0.0 | 0.00 | 1.0 | 1.00 | 1.0 |
| **family_size** | 2192.0 | 2.601277 | 0.907275 | 1.0 | 2.00 | 3.0 | 3.00 | 5.0 |
| **custm_tot_purchase($)** | 2192.0 | 613.523266 | 601.768054 | 8.0 | 71.00 | 406.0 | 1052.25 | 2525.0 |

# Data Visualisation :

Comparing Earning between a Parent and a non Parent in relation to their Education Levels :

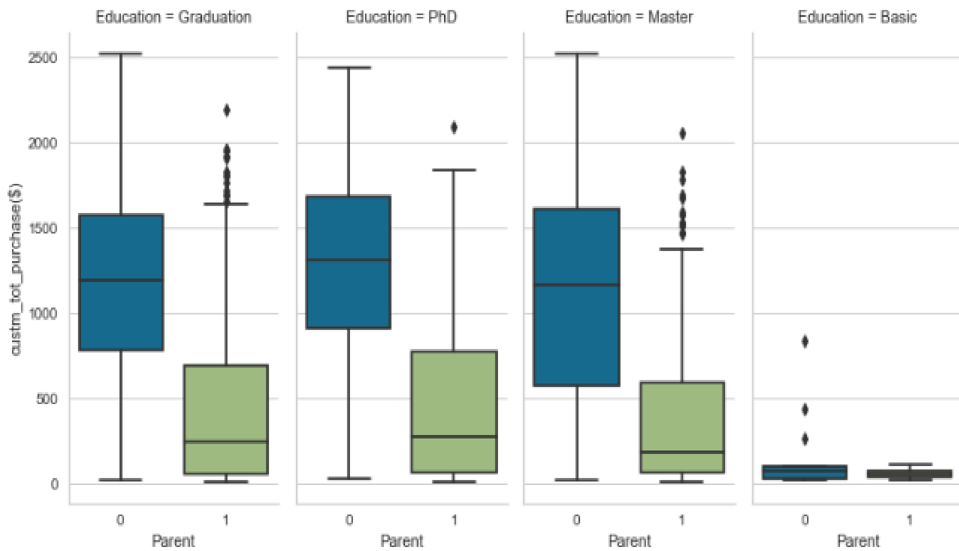
In [26]:
```
sns.catplot(x='Parent',y='Income($)',col='Education',kind='box',data=customer_data,aspect=0.5);
```




- Looking at the customer's income, those who are parents and have a degree(graduated) or a PhD have a slight difference of income as compared to non parents with the same education level.
- Customers who are not parents and their education level being Master earn more than those who are parents.
- Customers who are parents and have a basic education level earn more with a big difference than those who are not compared to other education levels.

## Customer Purchases between a Parent and a non-parent in relation to their education levels :

In [27]:
```
sns.catplot(x='Parent',y='custm_tot_purchase($)',col='Education',kind='box',data=customer_data,aspect=0.5);
```




## Showing the data in form of pivot table :

In [28]:
```
#Create a pivot table of Education in relation to Income, Total purchase and Customer's age
educ_pivot = pd.pivot_table(
    customer_data, index = "Education", values = ['Income($)','custm_tot_purchase($)','Customer_Age'], aggfunc = np.median
)
educ_pivot["Education"] = ["Basic", "Graduation", "Master", "PhD"]
educ_pivot = educ_pivot.reset_index(drop=True)
educ_pivot
```

Out[28]:

| | Customer_Age | Income($) | custm_tot_purchase($) | Education |
|---|---|---|---|---|
| 0 | 35.0 | 22390.0 | 54.0 | Basic |
| 1 | 44.0 | 52074.0 | 433.0 | Graduation |
| 2 | 44.0 | 50343.5 | 338.0 | Master |
| 3 | 46.5 | 54978.5 | 495.5 | PhD |

## Rescaling the pivot table :

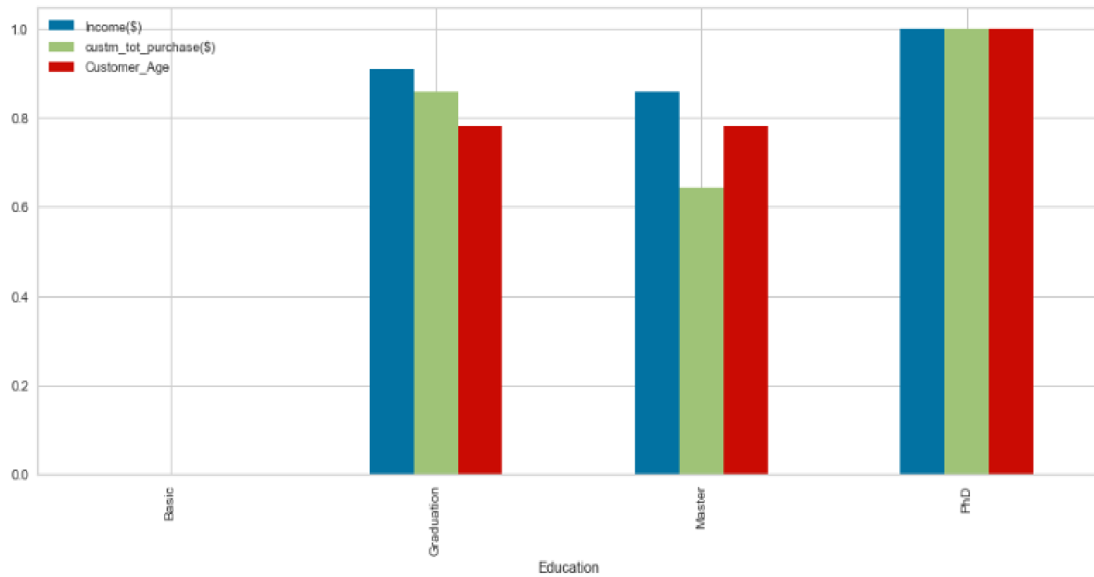
In [31]:
```
# Rescale the pivot table
# Get a scaler object
scaler = MinMaxScaler()

# Create a new dataframe for the scaled values
educ_normalized = educ_pivot[['Education', 'Income($)', 'custm_tot_purchase($)','Customer_Age']].copy()

# Normalize the numeric columns
educ_normalized[['Income($)', 'custm_tot_purchase($)','Customer_Age']] = scaler.fit_transform(educ_normalized[['Income($)', 'custm_tot_
```

## Plotting the normalized values :

```
In [32]: # Plot the normalized values
fig, ax = plt.subplots(figsize = [15, 6])
educ_normalized.plot(x='Education', y=['Income($)','custm_tot_purchase($)','Customer_Age'], kind='bar', ax = ax);
```



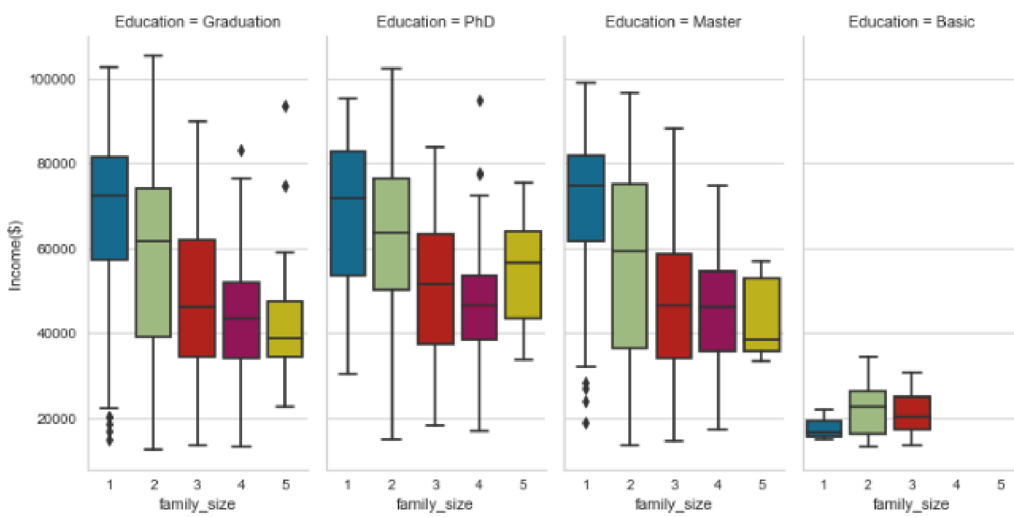

Results from above plot :

- Customers with basic education have the least income, the least purchase and are younger that other customers
- PhD customers have the highest income, highest purchase and are the oldest
- Graduate customers have relatively higher income and total number of purchase than Master customers.

## Size of the family in relation to Income and Education Levels :

In [33]:
```
sns.catplot(x='family_size',y='Income($)',col='Education',kind='box',data=customer_data,aspect=0.5)
```

Out[33]: <seaborn.axisgrid.FacetGrid at 0x298adba99d0>




## Customer Enrollment :

In [34]:
```
print('The recent customer enrollment date:',max(customer_data['Dt_Customer']))
print('The oldest customer enrollment date:',min(customer_data['Dt_Customer']))
```

```
The recent customer enrollment date: 2014-06-29 00:00:00
The oldest customer enrollment date: 2012-07-30 00:00:00
```

- The earliest customer enrollment was on 30th July 2012 with the recent being 29th June 2014. The dataset captures customer's purchases for two years. Hence the Age we have calculated based on the recent date of enrollment.

## Correlation Table :

In [35]:
```
customer_data.corr()
```

Out[35]:

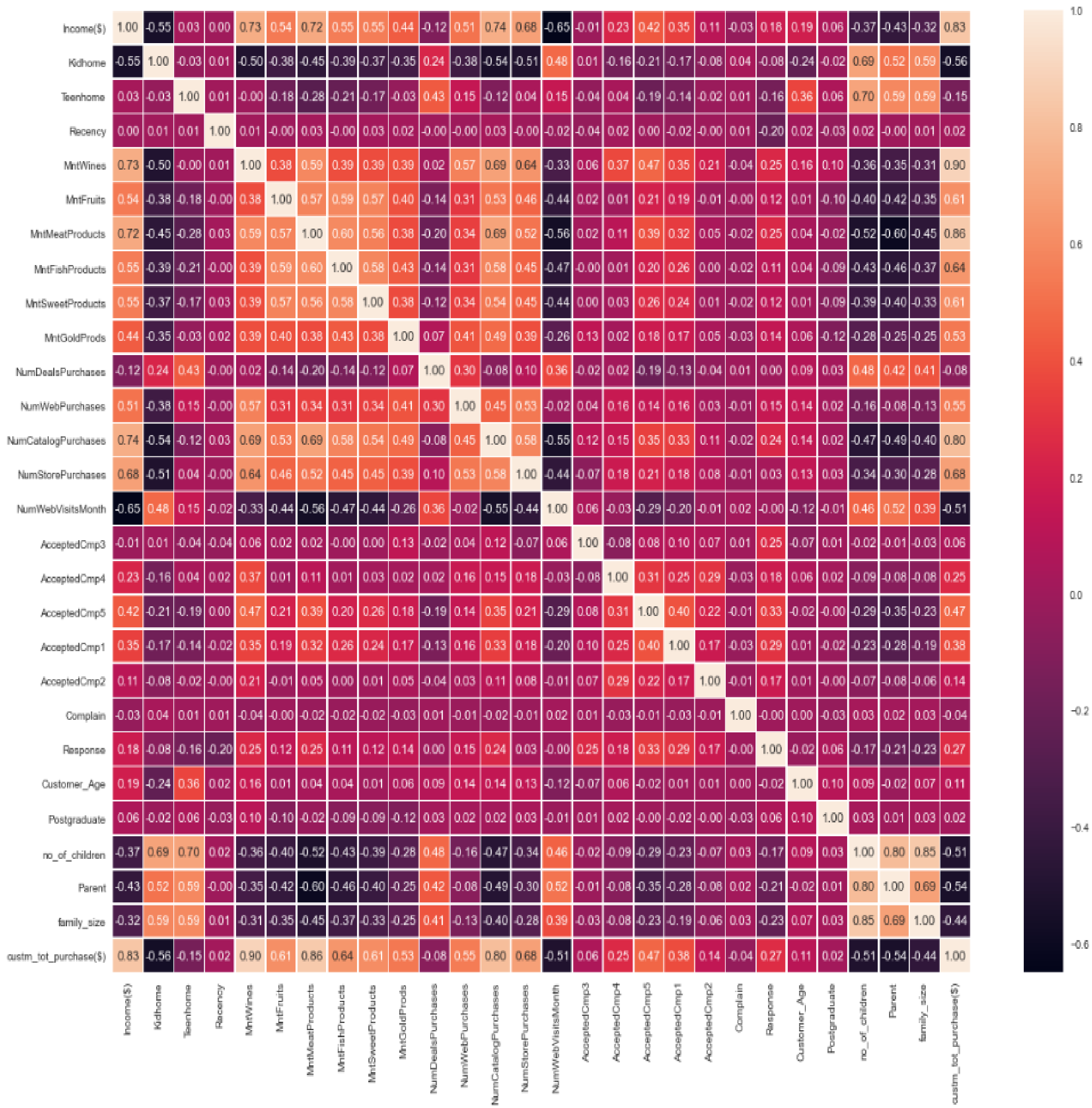
| | Income($) | Kidhome | Teenhome | Recency | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts | MntGoldProds | ... | Acceptec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Income($)** | 1.000000 | -0.545007 | 0.030643 | 0.003597 | 0.733222 | 0.541797 | 0.718365 | 0.554258 | 0.550828 | 0.435036 | ... | 0.3 |
| **Kidhome** | -0.545007 | 1.000000 | -0.033539 | 0.008403 | -0.502134 | -0.375997 | -0.454999 | -0.391133 | -0.374570 | -0.353730 | ... | -0.1 |
| **Teenhome** | 0.030643 | -0.033539 | 1.000000 | 0.013824 | -0.003000 | -0.182027 | -0.275574 | -0.211849 | -0.167450 | -0.032911 | ... | -0.1 |
| **Recency** | 0.003597 | 0.008403 | 0.013824 | 1.000000 | 0.014127 | -0.004758 | 0.025866 | -0.000111 | 0.025986 | 0.021408 | ... | -0.0 |
| **MntWines** | 0.733222 | -0.502134 | -0.003000 | 0.014127 | 1.000000 | 0.383687 | 0.591986 | 0.393106 | 0.385823 | 0.391816 | ... | 0.3 |
| **MntFruits** | 0.541797 | -0.375997 | -0.182027 | -0.004758 | 0.383687 | 1.000000 | 0.569261 | 0.592134 | 0.570484 | 0.395715 | ... | 0.1 |
| **MntMeatProducts** | 0.718365 | -0.454999 | -0.275574 | 0.025866 | 0.591986 | 0.569261 | 1.000000 | 0.596192 | 0.556292 | 0.379671 | ... | 0.3 |
| **MntFishProducts** | 0.554258 | -0.391133 | -0.211849 | -0.000111 | 0.393106 | 0.592134 | 0.596192 | 1.000000 | 0.582884 | 0.428967 | ... | 0.2 |
| **MntSweetProducts** | 0.550828 | -0.374570 | -0.167450 | 0.025986 | 0.385823 | 0.570484 | 0.556292 | 0.582884 | 1.000000 | 0.379624 | ... | 0.2 |
| **MntGoldProds** | 0.435036 | -0.353730 | -0.032911 | 0.021408 | 0.391816 | 0.395715 | 0.379671 | 0.428967 | 0.379624 | 1.000000 | ... | 0.1 |
| **NumDealsPurchases** | -0.116290 | 0.235440 | 0.427079 | -0.002745 | 0.023411 | -0.136140 | -0.195100 | -0.142408 | -0.119723 | 0.073897 | ... | -0.1 |
| **NumWebPurchases** | 0.510895 | -0.375107 | 0.148254 | -0.004807 | 0.567378 | 0.309415 | 0.338420 | 0.305254 | 0.341011 | 0.411459 | ... | 0.1 |
| **NumCatalogPurchases** | 0.738682 | -0.537872 | -0.116809 | 0.030488 | 0.686950 | 0.525085 | 0.692967 | 0.575428 | 0.535894 | 0.487912 | ... | 0.3 |
| **NumStorePurchases** | 0.679516 | -0.513723 | 0.042127 | -0.002760 | 0.636746 | 0.459198 | 0.517044 | 0.454250 | 0.452108 | 0.392718 | ... | 0.1 |
| **NumWebVisitsMonth** | -0.649510 | 0.480773 | 0.153719 | -0.023041 | -0.329597 | -0.440274 | -0.563202 | -0.468211 | -0.443565 | -0.261713 | ... | -0.2 |
| **AcceptedCmp3** | -0.009992 | 0.010165 | -0.044534 | -0.035832 | 0.062791 | 0.015589 | 0.022814 | -0.000093 | 0.002219 | 0.128102 | ... | 0.0 |
| **AcceptedCmp4** | 0.231098 | -0.162628 | 0.036625 | 0.018234 | 0.371673 | 0.007467 | 0.107736 | 0.013472 | 0.026513 | 0.021595 | ... | 0.2 |
| **AcceptedCmp5** | 0.422951 | -0.206276 | -0.194893 | 0.000100 | 0.471013 | 0.209930 | 0.389748 | 0.195641 | 0.259415 | 0.178575 | ... | 0.4 |
| **AcceptedCmp1** | 0.348265 | -0.173505 | -0.144229 | -0.020300 | 0.352685 | 0.193378 | 0.324649 | 0.258853 | 0.242371 | 0.169220 | ... | 1.0 |
| **AcceptedCmp2** | 0.110764 | -0.082238 | -0.016892 | -0.002145 | 0.205582 | -0.010953 | 0.045019 | 0.001170 | 0.009084 | 0.050648 | ... | 0.1 |
| **Complain** | -0.031976 | 0.036856 | 0.006803 | 0.005164 | -0.037887 | -0.003803 | -0.021775 | -0.020048 | -0.021432 | -0.030974 | ... | -0.0 |
| **Response** | 0.176478 | -0.081845 | -0.157444 | -0.198893 | 0.246665 | 0.124020 | 0.251440 | 0.109171 | 0.116821 | 0.142290 | ... | 0.2 |
| **Customer_Age** | 0.194869 | -0.236796 | 0.359420 | 0.018970 | 0.156106 | 0.009727 | 0.035259 | 0.036875 | 0.008767 | 0.055840 | ... | 0.0 |
| **Postgraduate** | 0.055068 | -0.015319 | 0.057420 | -0.032955 | 0.103748 | -0.096420 | -0.024004 | -0.086552 | -0.091957 | -0.118625 | ... | -0.0 |
| **no_of_children** | -0.367896 | 0.691428 | 0.698849 | 0.016007 | -0.361532 | -0.400669 | -0.524829 | -0.433059 | -0.389111 | -0.276944 | ... | -0.2 |
| **Parent** | -0.429039 | 0.516691 | 0.590584 | -0.002621 | -0.352510 | -0.419973 | -0.604021 | -0.459104 | -0.403358 | -0.252543 | ... | -0.2 |
| **family_size** | -0.316952 | 0.587050 | 0.594899 | 0.011078 | -0.306011 | -0.347668 | -0.454370 | -0.370616 | -0.333230 | -0.246056 | ... | -0.1 |
| **custm_tot_purchase($)** | 0.832812 | -0.562947 | -0.147165 | 0.020572 | 0.896417 | 0.614755 | 0.856815 | 0.643647 | 0.609951 | 0.534313 | ... | 0.3 |

28 rows × 28 columns

## Plotting the correlation table :

In [36]:

```python
plt.figure(figsize=(20,15))
sns.heatmap(data=customer_data.corr(),linewidth=.5,annot=True,fmt='.2f');
```



## Conclusion :

- We can see from the correlation heat map the values that have a strong positive correlation above 0.5 to 1.0 and strong negative strong correlation aboves -0.50 to -6.0
- There is a positive correlation between the Kids at home and the number of web visits per month as well as the number of deals purchased.
- When it comes to teens at home there is a positive correlation to the number of store purchases,number of web purchases and number of deals purchases as well as the amount spent on wine.
- With the objective being to increase customer purchases there is a strong positive correlation between the total customer purchases and the customer age,acceptance of the first campaign,number of deals purchases,fifth campaign,forth,campaign,amount of gold products purchased,amount of sweets purchased,amount of fish products purchased,number of web purchases,new store purchases,amount of fruit purchased,number of catalog purchases,amount of meat products purchased and the highest being amount of wine purchased.
- When it comes to campigns the campign5 has the highest correlation to the amount of wine purchased.
- The amount of meat products have a strong correlation with the customer total purchase.

## K-means Clustering :

The cleaned dataset was trained using the following clustering algorithms to group customers with similar patterns together into distinct clusters based on their inherent similarities or dissimilarities.

Clustering is a form of unsupervised learning, meaning that the algorithm identifies patterns and structure in the data without prior knowledge of the target labels or outcomes.

- K-Means
- DBSCANClusters
- Agglomerative Clustering

The following were carried out before training the clustering models with the clean dataset.

- The categorical features in the dataset were encoded using a LabelEncoder, in other to convert them to numerical features
- Some of the features of the dataset were transformed using the StandardScaler preprocessing technique, so that they have zero mean and unit variance
- We also carried out principal component analysis to reduce the dimension of the features
- Lastly, we used the elbow metrics to determine the number of clusters

## Encoding Numerical Features :

In [37]:
```
# numerical_features = [feature for feature in customer_data.columns if customer_data[feature].dtype != '0']
# categorical_features = [feature for feature in customer_data.columns if customer_data[feature].dtype == '0']
numerical_features = customer_data.select_dtypes("number").columns.to_list()
categorical_features = customer_data.select_dtypes("object").columns.to_list()
```

In [38]:
```
print(f'{numerical_features}\n*****\n{categorical_features}')
```

```
['Income($)', 'Kidhome', 'Teenhome', 'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGo
ldProds', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3', 'Accep
tedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Complain', 'Response', 'Customer_Age', 'Postgraduate', 'no_of_children', 'Pare
nt', 'family_size', 'custm_tot_purchase($)']
*****
['Education', 'Marital_Status']
```

In [39]:
```
LE = LabelEncoder()
for i in categorical_features:
    customer_data[i] = LE.fit_transform(customer_data[i])

print("All categorical features are now numerical")
```

```
All categorical features are now numerical
```

In [40]:
```
customer_data.head()
```

Out[40]:

| | Education | Marital_Status | Income($) | Kidhome | Teenhome | Dt_Customer | Recency | MntWines | MntFruits | MntMeatProducts | ... | AcceptedCmp1 | AcceptedCmp2 | Complain | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 58138.0 | 0 | 0 | 2012-09-04 | 58 | 635 | 88 | 546 | ... | 0 | 0 | 0 | |
| **1** | 1 | 2 | 46344.0 | 1 | 1 | 2014-03-08 | 38 | 11 | 1 | 6 | ... | 0 | 0 | 0 | |
| **2** | 1 | 1 | 71613.0 | 0 | 0 | 2013-08-21 | 26 | 426 | 49 | 127 | ... | 0 | 0 | 0 | |
| **3** | 1 | 1 | 26646.0 | 1 | 0 | 2014-02-10 | 26 | 11 | 4 | 20 | ... | 0 | 0 | 0 | |
| **4** | 3 | 1 | 58293.0 | 1 | 0 | 2014-01-19 | 94 | 173 | 43 | 118 | ... | 0 | 0 | 0 | |

5 rows × 31 columns

In [41]:
```
customer_data.drop(columns = ["Dt_Customer"], inplace = True)
```

In [42]:
```
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Complain', 'Response']
```

In [43]:
```
for f in numerical_features:
    if f not in cols_del:
        sc = StandardScaler()
        customer_data[f] = sc.fit_transform(customer_data[f].values.reshape(-1, 1))
```

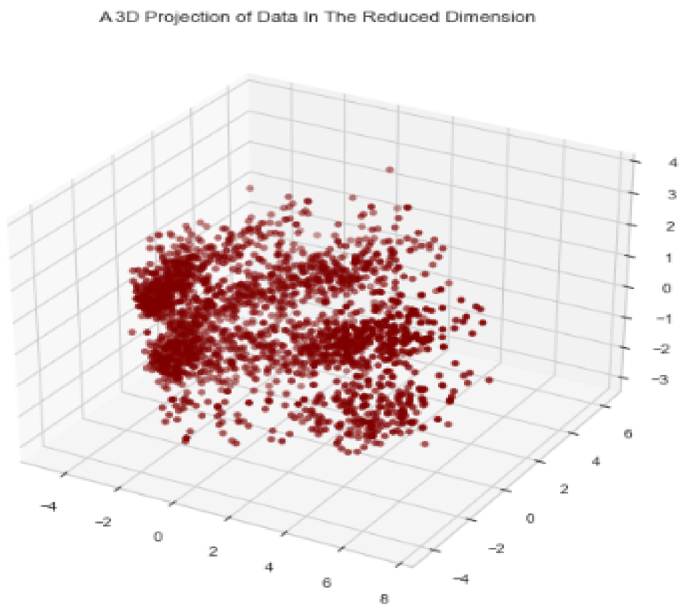
In [44]:
```
#Initiating PCA to reduce dimension aka features to 3
pca = PCA(n_components = 3)
pca.fit(customer_data)
PCA_ds = pd.DataFrame(pca.transform(customer_data), columns = (["col1", "col2", "col3"]))
PCA_ds.describe().T
```

Out[44]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **col1** | 2192.0 | 2.374419e-16 | 2.923410 | -4.759140 | -2.602898 | -0.794962 | 2.460639 | 7.540830 |
| **col2** | 2192.0 | -3.545420e-17 | 1.714234 | -4.330504 | -1.334383 | -0.150277 | 1.224769 | 6.336028 |
| **col3** | 2192.0 | -6.493184e-17 | 1.329726 | -2.983159 | -1.227035 | 0.356729 | 1.112597 | 3.729379 |

## 3-D projection of reduced data ;

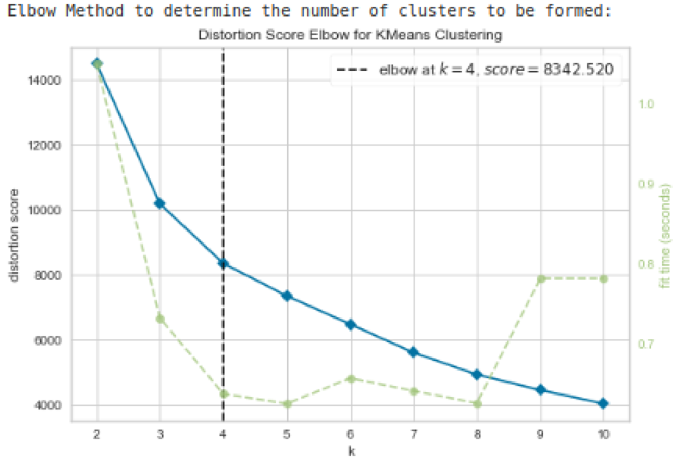
In [45]:
```
#A 3D Projection of Data In The Reduced Dimension
x = PCA_ds["col1"]
y = PCA_ds["col2"]
z = PCA_ds["col3"]
fig = plt.figure(figsize=[10, 8])
ax = fig.add_subplot(111, projection = "3d")
ax.scatter(x, y, z, c = "maroon", marker = "o")
ax.set_title("A 3D Projection of Data In The Reduced Dimension")
plt.show()
```




## Using Elbow method to find out the number of clusters to make :

In [46]:
```
#Quick examination of elbow to find the number of clusters to make
print('Elbow Method to determine the number of clusters to be formed: ')
Elbow_M = KElbowVisualizer(KMeans(), k = 10)
Elbow_M.fit(PCA_ds)
Elbow_M.show()
```

```
Elbow Method to determine the number of clusters to be formed:
```

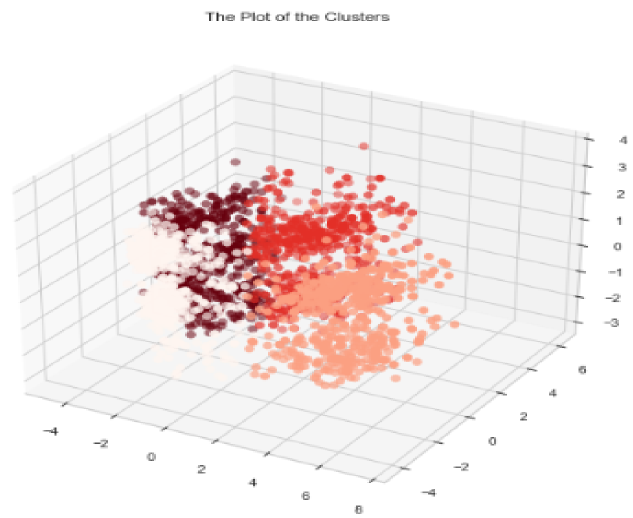



Out[46]:
```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

## Using K-means :

In [47]:
```
kmeans = KMeans(n_clusters = 4)
yhat_kmeans = kmeans.fit_predict(PCA_ds)
PCA_ds["KmeansClusters"] = yhat_kmeans
customer_data["KmeansClusters"] = yhat_kmeans
```

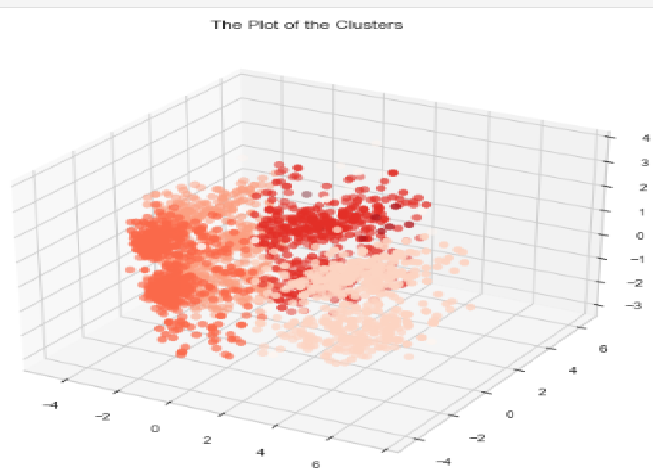
In [48]:
```
#Plotting the clusters
fig = plt.figure(figsize=[10, 8])
ax = fig.add_subplot(111, projection = "3d", label = "bla")
ax.scatter(x, y, z, s = 40, c = PCA_ds["KmeansClusters"], marker = "o", cmap = "Reds")
ax.set_title("The Plot of the Clusters")
plt.show()
```




## Using DBSCAN :

In [49]:
```
dbscan = DBSCAN(eps = 1)
yhat_dbscan = dbscan.fit_predict(PCA_ds)
PCA_ds["DBSCANClusters"] = yhat_dbscan
customer_data["DBSCANClusters"] = yhat_dbscan
```

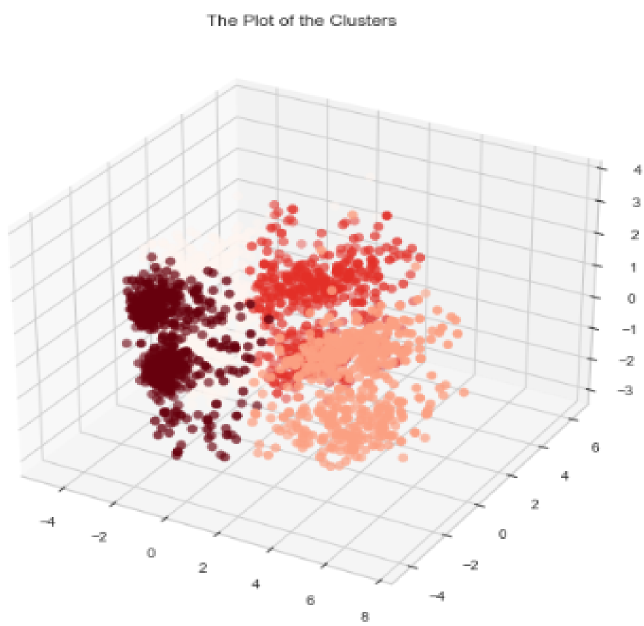
In [50]:
```
#Plotting the clusters
fig = plt.figure(figsize=[10, 8])
ax = fig.add_subplot(111, projection = "3d", label = "bla")
ax.scatter(x, y, z, s = 40, c = PCA_ds["DBSCANClusters"], marker = "o", cmap = "Reds")
ax.set_title("The Plot of the Clusters")
plt.show()
```



## Using AgglomerativeClustering;

In [51]:
```
AC = AgglomerativeClustering(n_clusters = 4)
#Fit the model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
PCA_ds["AggClusters"] = yhat_AC
#Adding the clusters features to the original dataframe
customer_data["AggClusters"] = yhat_AC
```

In [52]:
```
#Plotting the clusters
fig = plt.figure(figsize=[10, 8])
ax = fig.add_subplot(111, projection = "3d", label = "bla")
ax.scatter(x, y, z, s = 40, c = PCA_ds["AggClusters"], marker = "o", cmap = "Reds")
ax.set_title("The Plot of the Clusters")
plt.show()
```




## Conclusion :

The summary of the model is as follows:

- The optimal value for clusters is 4
- The distortion score is 8342.520

From the plots we can see that there are:

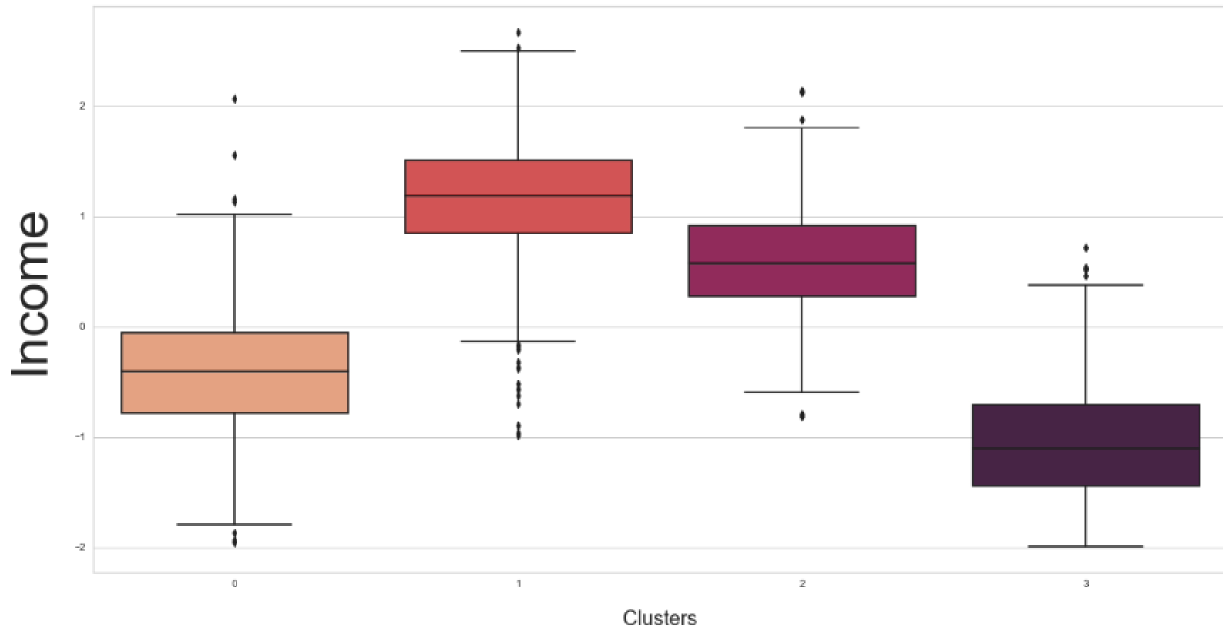
- Good metric results for Agglomerative, DBSCAN and K-means Clustering algorithms.
- Best Model: AgglomerativeClustering(n_clusters=4).

The AgglomerativeClustering model achieved the highest level of performance and was used to generate the clusters

## Plotting Income Column :

In [55]:
```
#Income
plt.figure(figsize=(20,10))

sns.boxplot(data=customer_data, x='AggClusters', y = 'Income($)',palette='rocket_r');
plt.xlabel('Clusters', fontsize=20, labelpad=20)
plt.ylabel('Income', fontsize=50, labelpad=20);
```



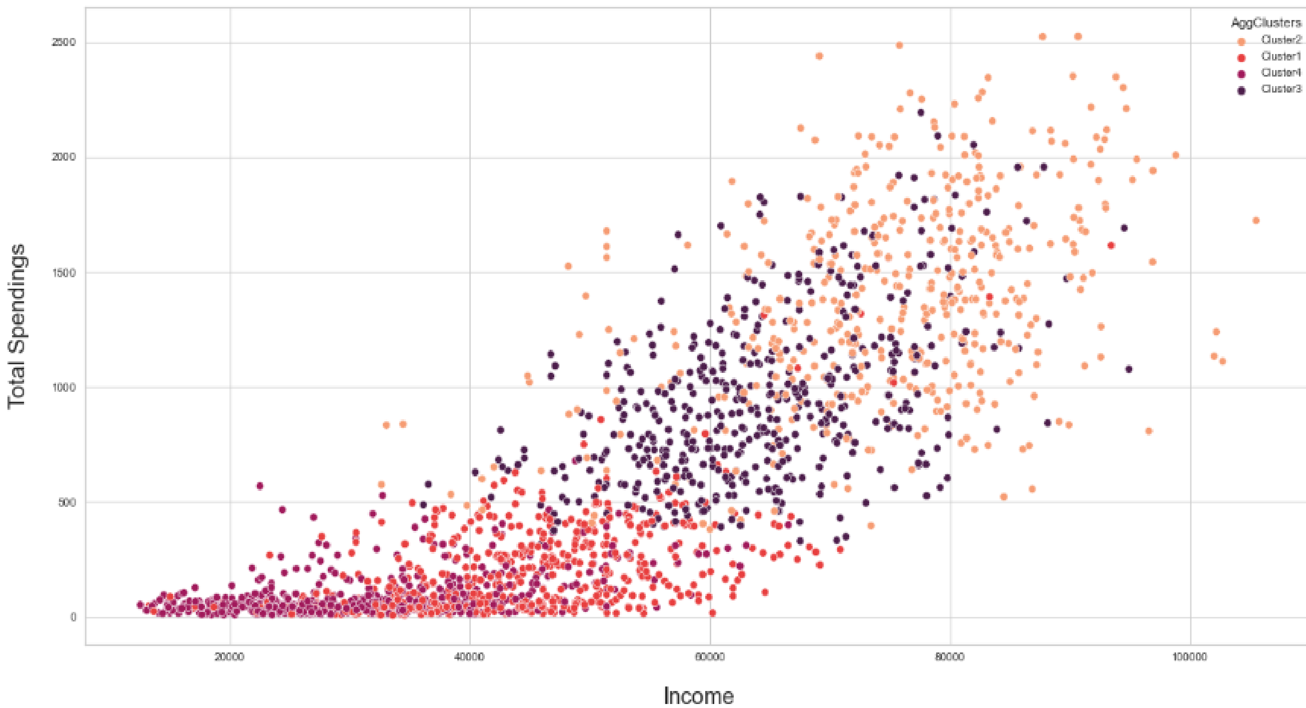

- We can see from the above plot the relation between clusters and income column .
- So , we can divide the clusters on the basis of the income .
- Using scatter plotting to understand the clusters behavior .

## Scatter Plot :

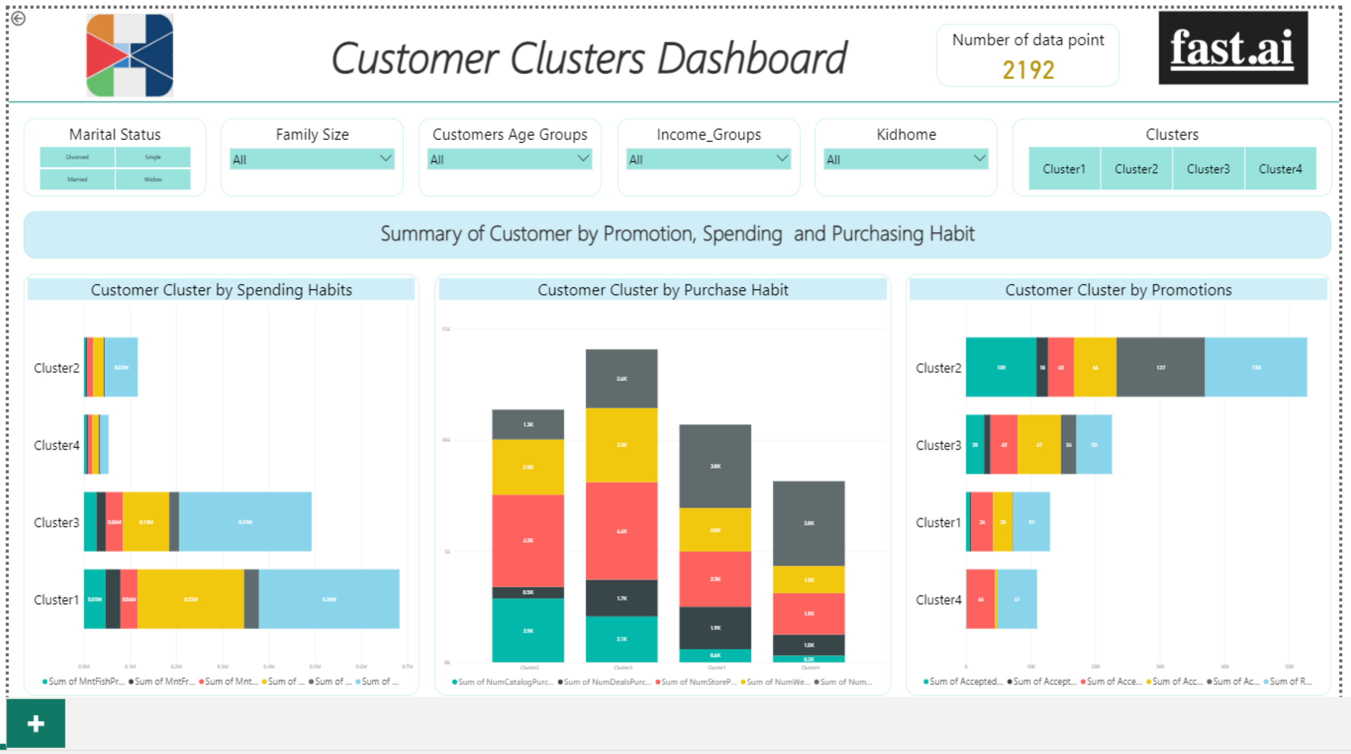
In [56]:
```
customer_data.AggClusters = customer_data.AggClusters.replace({0: 'Cluster1',
                                                               1: 'Cluster2',
                                                               2: 'Cluster3',
                                                               3: 'Cluster4'})

customer_new['AggClusters'] = customer_data.AggClusters
```

In [57]:
```
plt.figure(figsize=(20,10))
sns.scatterplot(data=customer_new, x='Income($)', y='custm_tot_purchase($)', hue="AggClusters", palette='rocket_r');
plt.xlabel('Income', fontsize=20, labelpad=20)
plt.ylabel('Total Spendings', fontsize=20, labelpad=20);
```



# Model Deployment :




## Summary of each cluster :

Cluster 1 - Traits :

- People in cluster 1 earn little and spend little .
- They have either 0 kids or more kids .
- They are mostly married people .
- They are mostly postgraduate students .
- They have a family size of about 2-5 .
- They have at least one teen at home .
- They are within the age of 30-68 .
- They have the third least total purchase .
- Conclusion : People in cluster one earn little and spend little (Low Spenders) .

Cluster 2 - Traits :

- People in cluster 2 earn highest and spend highest .
- They have no kids at home .
- Majority of people in cluster 2 are married while others are single,divorced or widowed .
- Majority fall in the post-graduate category .
- They have a family size of 1-2 .
- They do not have a teen at home .
- Their age ranges from 19-73 .
- They have the highest total purchase .
- Conclusion : People in cluster two earn highest and spend highest. (High Spenders) .

Cluster 3 - Traits :

- People in cluster 3 earn high and spend high.
- They have at most one kid at home .
- Majority of the people in cluster three are married .
- Majority fall in the postgraduate category .
- They have a family size of 2-4 .
- Majority have one teen at home .
- Their age range is from 24-67 .
- They have the second highest total purchase .
- Conclusion : People in cluster three earn high and spend high. (Average Spenders)

Cluster 4 - Traits :

- People in cluster four earn lowest and spend lowest .
- They have at most one kid at home .
- Majority of the people in cluster four  are married and single .
- They fall in the postgraduate and basic education .
- They have a family size of 1-3 .
- They have no teen at home .
- Their age range is from 24-74 .
- They have the least total purchase .
- Conclusion : People in cluster four earn lowest and spend lowest.(Minimal Spenders)

Answering Business Question

**What are the Characteristics of Customers?**

.Customers are mostly married.

.They are mostly without kids.

. They are educated. Majority of the customers are middle aged between the age of 33 - 68 and majority wants to have at most one kid.

.They earn between 15000 - 98000..

**What are the Shopping habits of Customers?**

.Customers spend more on Mntwines and MntMeatproducts in the last two years.

.Majority of the purchase were made from instore and web purchase.

.Customers without kids spend more.

.Those that are married spend more than those that are single.

.Those who are graduate degree holders spend more too.

.Customer who are middle - aged have spent more than any other age group.

**Are there products that need more promotions ?**

.Products like mntsweetpotato, MntFishProducts, MntFruits needs more promotional campaigns. However, mntGold also have low purchase but every category of customers can't afford it.

.The company should focus on promotional Campaigns like advertising the importance of these products from the health benefits they can provide to customers and also provide discounts on these products to further drive sales.

**How can Marketing be made more effective?**

Give discounts on products with low purchases.

Customers who spend higher can be given discounts on products they spend more alongside low purchase products.

.Drive traffic towards the web by giving discounts to customers who register and make purchases via the web.

## Project Summary :

In conclusion, the clustering analysis provided valuable insights into customer personalities and identified four distinct groups that could be targeted with specific marketing strategies.

The challenges we faced in the project is the scheduling of meetings because coordinating group meetings can be difficult when members are in different time zones.

Each member needed to adjust their schedules to accommodate the group and the team leadership had to find a time that works for everyone .

Our recommendations are:

- Future work using alternative clustering algorithms
- Incorporating additional data sources to further refine the customer segments.

The project, if properly utilized, would contribute to the development of a more data-driven and customer-focused approach to marketing, which can help businesses improve customer satisfaction and increase revenue.

- Customer segmentation: The clustering analysis identified four clusters, which can be used to segment customers and tailor marketing strategies to their specific needs and preferences.
- Improved marketing strategies: By understanding the personality traits of different customer segments, businesses can develop more effective marketing strategies that resonate with their target audience.
- Customer satisfaction: By tailoring marketing strategies to specific personality groups, businesses can improve customer satisfaction by delivering more relevant and personalized marketing messages.
- Data-driven decision making: The project demonstrated the value of using data-driven approaches to gain insights into customer behavior and preferences, which can help inform business decisions and improve overall performance.

## Important Links :

- The link to github notebook is :

  https://github.com/John-ChuOnyido/Fastai/blob/main/Customer%20Personality%20Analysis.ipynb

# Thank you!

Documentation by : Harsh Pant

Team : FastAI