Marketing Analytics Exploratory Data Analysis (EDA)

As a food company aiming to maximize profits in the upcoming direct marketing campaign scheduled for next month, we conducted a pilot campaign involving 2,206 customers. Those who purchased the offer were clearly labeled.

Our team's objective is to comprehend the characteristic features of these customers. As part of this analysis, we will delve into the data

Imports

** Import libraries used in the EDA **

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Get the Data

We will work with the ifood_df.csv from the company. It has income, number of kids and teenagers in a family, different expenses, marital and educational status.

```
mkt_data=pd.read_csv("ifood_df.csv")
mkt_data.head(5)
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProc
0	58138.0	0	0	58	635	88	546	
1	46344.0	1	1	38	11	1	6	
2	71613.0	0	0	26	426	49	127	
3	26646.0	1	0	26	11	4	20	
4	58293.0	1	0	94	173	43	118	

5 rows × 39 columns

Next, shape of the dataframe

```
mkt_data.shape
```

```
(2205, 39)
```

Dataframe contains, 2205 rows and 39 columns.

info() of the data...

```
mkt_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 39 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Income                2205 non-null   float64
 1   Kidhome               2205 non-null   int64
 2   Teenhome              2205 non-null   int64
 3   Recency               2205 non-null   int64
 4   MntWines              2205 non-null   int64
 5   MntFruits             2205 non-null   int64
 6   MntMeatProducts       2205 non-null   int64
 7   MntFishProducts       2205 non-null   int64
 8   MntSweetProducts      2205 non-null   int64
 9   MntGoldProds          2205 non-null   int64
```

```
 10  NumDealsPurchases     2205 non-null   int64
 11  NumWebPurchases       2205 non-null   int64
 12  NumCatalogPurchases   2205 non-null   int64
 13  NumStorePurchases     2205 non-null   int64
 14  NumWebVisitsMonth     2205 non-null   int64
 15  AcceptedCmp3          2205 non-null   int64
 16  AcceptedCmp4          2205 non-null   int64
 17  AcceptedCmp5          2205 non-null   int64
 18  AcceptedCmp1          2205 non-null   int64
 19  AcceptedCmp2          2205 non-null   int64
 20  Complain              2205 non-null   int64
 21  Z_CostContact         2205 non-null   int64
 22  Z_Revenue             2205 non-null   int64
 23  Response              2205 non-null   int64
 24  Age                   2205 non-null   int64
 25  Customer_Days         2205 non-null   int64
 26  marital_Divorced      2205 non-null   int64
 27  marital_Married       2205 non-null   int64
 28  marital_Single        2205 non-null   int64
 29  marital_Together      2205 non-null   int64
 30  marital_Widow         2205 non-null   int64
 31  education_2n Cycle    2205 non-null   int64
 32  education_Basic       2205 non-null   int64
 33  education_Graduation  2205 non-null   int64
 34  education_Master      2205 non-null   int64
 35  education_PhD         2205 non-null   int64
 36  MntTotal              2205 non-null   int64
 37  MntRegularProds       2205 non-null   int64
 38  AcceptedCmpOverall    2205 non-null   int64
dtypes: float64(1), int64(38)
memory usage: 672.0 KB
```

All the columns are of integer type except income which is of Float. All columns have non-null values as count is 2205 in each column.

Statistical Analysis

describe() on the data...

```
mkt_data.describe()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	Mnt
count	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	2205.000000	
mean	51622.094785	0.442177	0.506576	49.009070	306.164626	26.403175	
std	20713.063826	0.537132	0.544380	28.932111	337.493839	39.784484	
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	35196.000000	0.000000	0.000000	24.000000	24.000000	2.000000	
50%	51287.000000	0.000000	0.000000	49.000000	178.000000	8.000000	
75%	68281.000000	1.000000	1.000000	74.000000	507.000000	33.000000	
max	113734.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	

8 rows × 39 columns

On average:

People earned 51K, with the highest income in the dataset being 114K. People spent 562, with the highest expenses in the dataset reaching 2491.

Unique Values...

```
mkt_data.nunique().sort_values(ascending=False)
```

```
Income                 1963
MntRegularProds         974
MntTotal                897
MntWines                775
Customer_Days           662
```

```
MntMeatProducts          551
MntGoldProds             212
MntFishProducts          182
MntSweetProducts         176
MntFruits                158
Recency                  100
Age                       56
NumWebVisitsMonth         16
NumDealsPurchases         15
NumWebPurchases           15
NumStorePurchases         14
NumCatalogPurchases       13
AcceptedCmpOverall         5
Teenhome                   3
Kidhome                    3
AcceptedCmp1               2
marital_Together           2
AcceptedCmp3               2
education_PhD              2
education_Master           2
education_Graduation       2
education_Basic            2
education_2n Cycle         2
marital_Widow              2
marital_Single             2
marital_Married            2
marital_Divorced           2
AcceptedCmp4               2
AcceptedCmp5               2
Response                   2
Complain                   2
AcceptedCmp2               2
Z_Revenue                  1
Z_CostContact              1
dtype: int64
```

Check Null Values

```
mkt_data.isna().sum()
```

```
Income                  0
Kidhome                 0
Teenhome                0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
Age                     0
Customer_Days           0
marital_Divorced        0
marital_Married         0
marital_Single          0
marital_Together        0
marital_Widow           0
education_2n Cycle      0
education_Basic         0
education_Graduation    0
education_Master        0
education_PhD           0
MntTotal                0
MntRegularProds         0
AcceptedCmpOverall      0
dtype: int64
```

There are no null values in the dataset

Duplicate Rows...

```
mkt_data_dup=mkt_data[mkt_data.duplicated()]
print("you have {} duplicate rows".format(len(mkt_data_dup)))
```

```
you have 184 duplicate rows
```

Drop Duplicates...

```
# Drop 184 duplicate rows
mkt_data.drop_duplicates(keep=False, inplace=True)
```

Data Preparation

In the provided data, marital status is presented across 5 different columns. Below, I have consolidated them into one column to streamline the data structure. The following steps were taken:

Converted the column to a string. Replaced '1' with the corresponding numerical categorical value. Created a new column named marital_status. Summed up all columns with marital status information. Replaced the categorical numerical values with strings such as 'Married', 'Divorced', etc.

```
#First, Change Column data type to string and then replace 1 with different number & 0 with blank.
mkt_data['marital_Married']=mkt_data['marital_Married'].astype(str).replace({'1':'5','0':''})
mkt_data['marital_Single']=mkt_data['marital_Single'].astype(str).replace({'1':'4','0':''})
mkt_data['marital_Together']=mkt_data['marital_Together'].astype(str).replace({'1':'3','0':''})
mkt_data['marital_Widow']=mkt_data['marital_Widow'].astype(str).replace({'1':'2','0':''})
mkt_data['marital_Divorced']=mkt_data['marital_Divorced'].astype(str).replace({'0':''})
#Now all columns contain different numbers for different marital status, lets join them in one column.
mkt_data['marital_status']=mkt_data["marital_Widow"]+mkt_data['marital_Together']+mkt_data['marital_Single']+mkt_data['marital_Married']+mkt
#Next, we map numbers into different categorical values.
mkt_data['marital_status']=mkt_data['marital_status'].map({'1':'Divorced', '2':'Widow','3':'Together','4':'Single','5':'Married'})
```

Let's do same operation, as above, for education columns.

```
mkt_data['education_2n Cycle']=mkt_data['education_2n Cycle'].astype(str).replace({'0':''})
mkt_data['education_Basic']=mkt_data['education_Basic'].astype(str).replace({'1':'2','0':''})
mkt_data['education_Graduation']=mkt_data['education_Graduation'].astype(str).replace({'1':'3','0':''})
mkt_data['education_Master']=mkt_data['education_Master'].astype(str).replace({'1':'4','0':''})
mkt_data['education_PhD']=mkt_data['education_PhD'].astype(str).replace({'1':'5','0':''})
mkt_data['education_level']=mkt_data['education_2n Cycle']+mkt_data['education_Basic']+mkt_data['education_Graduation']+mkt_data['education_
mkt_data['education_level']=mkt_data['education_level'].map({'1':'2n Cycle','2':'Basic','3':'Graduation','4':'Master','5':'PhD'})
```

for feature engineering, we join KidHome And TeenHome to find out number of children in a home

```
mkt_data['kids']=mkt_data['Kidhome']+mkt_data['Teenhome']
```

Next, we drop all unnecessary columns to make dataset simple.

```
mkt_data.drop(['education_2n Cycle','education_Basic','education_Graduation','education_Master','education_PhD','marital_Widow','marital_Tog
```

	Income	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetPr
0	58138.0	58	635	88	546	172	
1	46344.0	38	11	1	6	2	
2	71613.0	26	426	49	127	111	
3	26646.0	26	11	4	20	10	
4	58293.0	94	173	43	118	46	

with data restructure, dataset now contains 30 columns instead of 39.

EDA & Visualizations

Univariate Analysis

Univariate analysis is the simplest form of analysis where we explore a single variable. Univariate analysis is performed on both Numerical and categorical variables differently as plotting uses different plots.

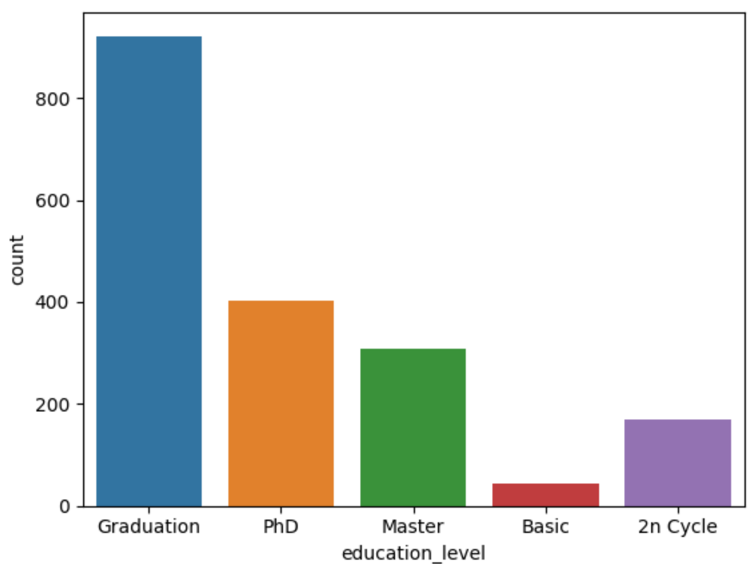
Customer Profile Analysis

Categorical Variables:

Lets start with categorical variables.

```
sns.countplot(x="education_level", data=mkt_data)
plt.show()
```

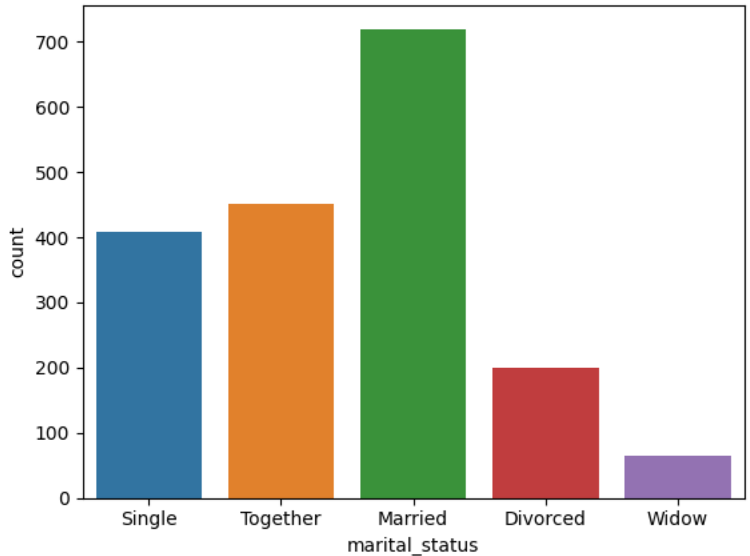
```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
```



In the given dataset, people are mainly graduated followed by PHD and Master degree thats is held by very less people in comparison to number of people who are graduated.

```
sns.countplot(x="marital_status", data=mkt_data)
plt.show()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
```

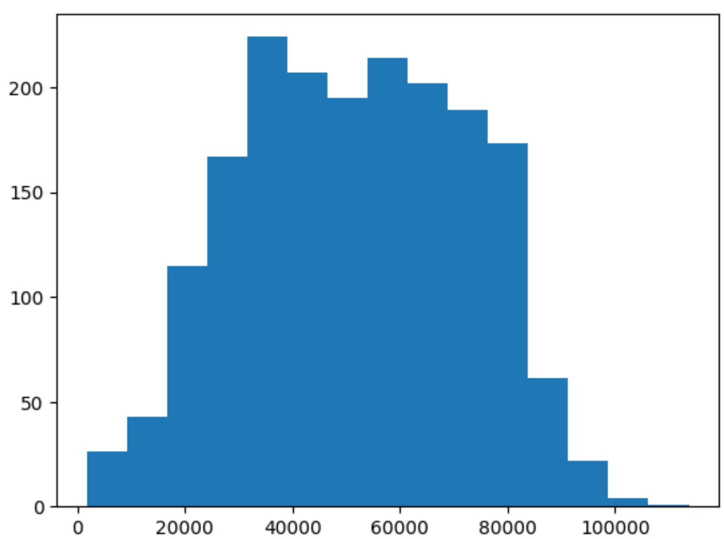


The majority of individuals in the dataset are married, followed by a significant number of people who are in relationships ('Together') and those who are single. There are relatively few individuals who are either divorced or widowed in this dataset.

Numerical Data: Analyzing numerical data is crucial because comprehending the distribution of features aids in the further processing of data. Often, inconsistencies arise within numerical data, making it essential to explore numerical variables. This exploration allows for a deeper understanding of the underlying patterns, trends, and potential outliers, facilitating more informed decision-making in data processing and analysis.

Let's analyze what is income distribution in our dataset.

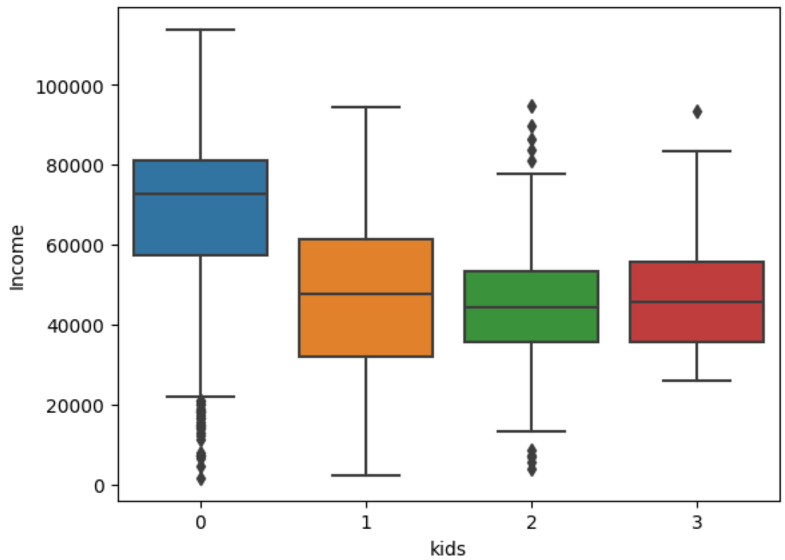
```
plt.hist(mkt_data["Income"], bins=15)
plt.show()
```



The histogram indicates that most people have incomes between 3000-8000. Using a box plot with the seaborn library, we examine income distribution and assess if the number of children influences it visually.

```
sns.boxplot(y=mkt_data["Income"], x=mkt_data["kids"])
plt.show()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
```



The graph provides valuable insights:

1. Individuals without children tend to have higher salaries, contrary to the general assumption that more kids would require more income.
2. As the number of children increases, people tend to have lower incomes. For instance, those with one child typically earn more than those with two or three children.

Correlation

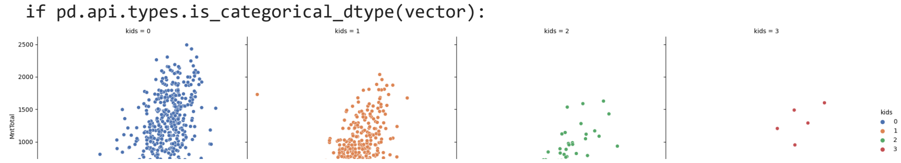
Bi-Variate Analysis

We have studied various plots for exploring single categorical and numerical data. Now, let's delve into Bivariate Analysis, employed to examine the relationship between two different variables, often focusing on correlation. This step is crucial in our overarching goal of understanding variable relationships to construct a robust model. Additionally, when analyzing more than two variables simultaneously, it is termed as Multivariate Analysis. We will explore different plots for both Bivariate and Multivariate Analysis to gain a comprehensive understanding of the data.

Below, we will assess whether there is any relationship between income and expenses. Additionally, we will explore if expenses are influenced by the number of children in a household. This Bivariate Analysis aims to uncover connections and dependencies between these key variables.

```
# i have used seaborn relplot to plot Income and MntTotal relationship using scatter plot.
#As I also wanted to see if this relationship changes with number of kids at home, replots allow to make subplots of a graph.
sns.relplot(x="Income", y="MntTotal", data=mkt_data, col="kids",hue="kids", kind="scatter",palette="deep")
plt.show()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
```



Income, total spending, and the number of kids show a negative correlation. People tend to earn and spend more when they have no kids, and expenses decrease as the number of kids increases. Those with three kids spend less than those with no kids or one kid.

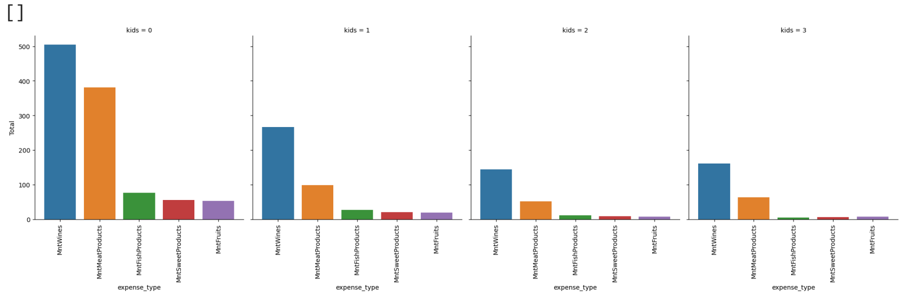
Next, we'll explore:

1. Whether individuals have specific preferences in spending on items like wine or fish, or if their expenses are uniform across all categories.
2. How having kids may influence priorities and habits, resulting in varied expenses across different categories. This analysis aims to uncover patterns in spending behaviors and identify any shifts in priorities related to the presence of children.

```
# Our dataset contains these expenses in different columns, first group the items by kids and sum expenses in different category.then restru
expenses = mkt_data.groupby(["kids"])[["MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts"]].mean().unstack().

#Here, i have used seaborn cat plot as i wanted to make subplots for number of kids.
chart=sns.catplot(data=expenses.sort_values(by="Total",ascending=False),x="expense_type",y="Total",kind="bar",legend=True, col="kids")
chart.set_xticklabels(rotation=90)
plt.plot()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
[]
```



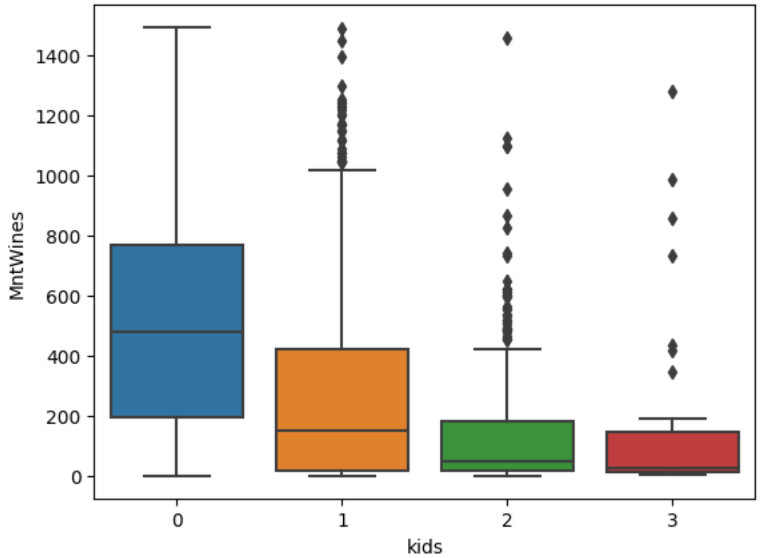
In this analysis, it appears that individuals, regardless of whether they have children or not, tend to spend more on wine and meat products compared to fish, sweets, and fruits. This suggests that having children does not significantly alter spending priorities in terms of expense categories.

Moreover, the previous observation holds true here: individuals with no kids tend to earn more and consequently spend more, consistent with the patterns observed in these graphs.

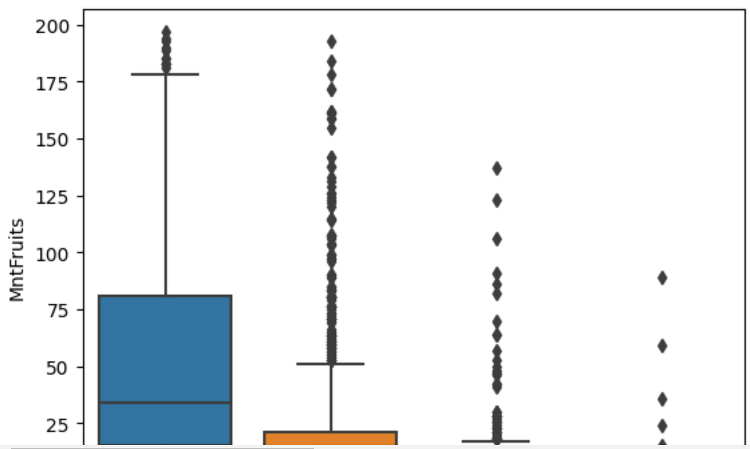
Above we saw, different expenses in each category of people having 0,1,2,3 kids. Now, let's compare people expenses with 0,1,2,3 kids under each expense category.

```
for col in mkt_data.columns:
    if 'Mnt' in col:
        sns.boxplot(x="kids", y=col, data=mkt_data)
        plt.show()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldc
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldc
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldc
  if pd.api.types.is_categorical_dtype(vector):
```



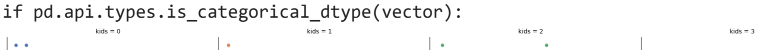

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldc
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldc
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldc
  if pd.api.types.is_categorical_dtype(vector):
```



Let's assess if people with more kids tend to buy more deals, possibly due to lower income. We'll plot the distribution of income against the number of deals purchased (NumDealsPurchases) for each category of the number of kids.

```
# We use, seaborn rel plot in order to make subplots for number of kids.
sns.relplot(data=mkt_data, x="Income", y="NumDealsPurchases", col="kids", hue="kids",palette="deep")
plt.show()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
```

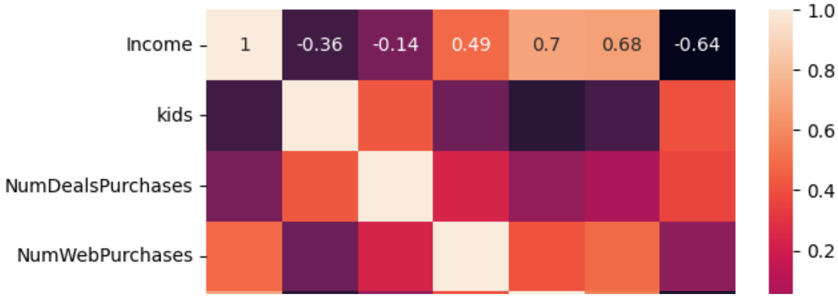


It's intriguing to observe that individuals with no kids tend to purchase fewer deals, despite earning more and spending more. Conversely, those with 1 or 2 kids, earning less and spending less, actively seek out deals. Meanwhile, individuals with 3 or more kids, who have lower income and spend very little, also buy fewer deals. This insight suggests a nuanced relationship between family size, income, spending behavior, and deal purchases.

We'll map the relationships among purchase channels, income, the number of kids, and the number of deals bought to gain a comprehensive understanding of their interplay.

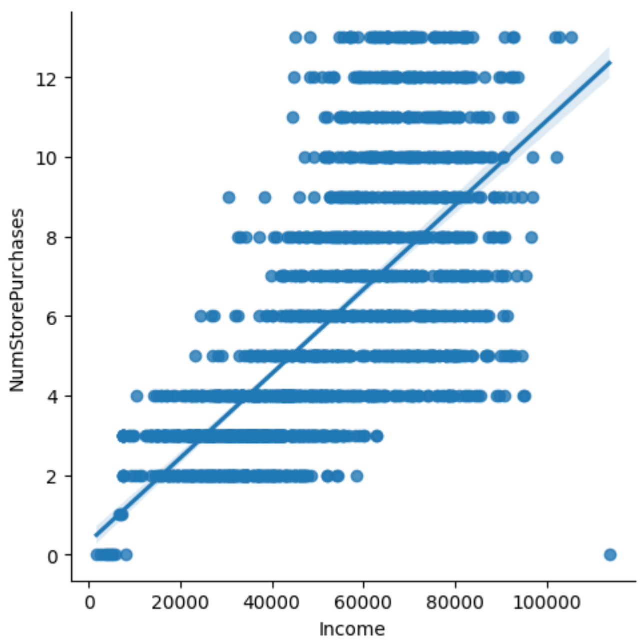
```
#For this, we use seaborn co relation heat map to check which elements have stronger relationship & then we will plot those elements seperat
mkt_data1=mkt_data[["Income","kids","NumDealsPurchases","NumWebPurchases","NumCatalogPurchases","NumStorePurchases","NumWebVisitsMonth"]]
sns.heatmap(mkt_data1.corr(), annot=True)
```

```
<Axes: >
```



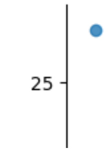
In the heatmap, a coefficient closer to 1 indicates a strong relationship between two elements. Income is strongly related to the number of catalog and store purchases. We will plot these to further explore their relationships. Given the earlier observation that people with no kids, who earn more, also spend more on wine, the positive correlation between income and expenses on wine aligns with the positive correlation observed between income and catalog/store purchases. This suggests that wine is primarily purchased in stores.


```
sns.lmplot(data=mkt_data, x="Income", y="NumStorePurchases")
plt.show()
```



The graph illustrates a positive correlation, indicating that individuals with higher income tend to make more store purchases.

```
sns.lmplot(data=mkt_data, x="Income", y="NumCatalogPurchases")
plt.show()
```

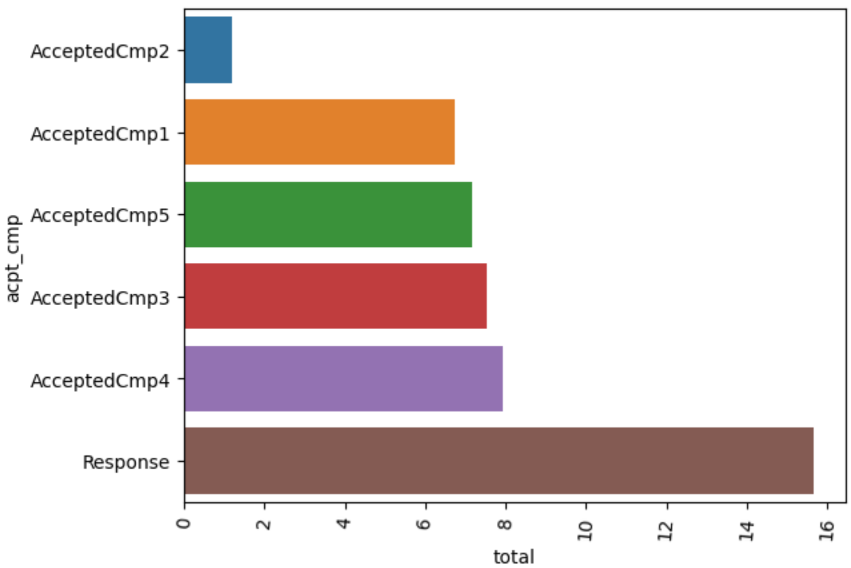


People with higher income also make more catalogue purchases.

Let's analyze the accepted campaign and response data. We'll plot the percentage of each campaign accepted to determine which one performed better. Additionally, we'll examine the number of people who responded to the last campaign.

```
##### First, we calculate the % of each campaign accepted and then we plot them in a bar graph.
cmp_success=((mkt_data[["AcceptedCmp3","AcceptedCmp4","AcceptedCmp5","AcceptedCmp1","AcceptedCmp2", "Response"]].sum(axis=0)/ mkt_data[["Acc
sns.barplot(x="total", y="acpt_cmp", data=cmp_success)
plt.xticks(rotation=85)
plt.show()
```

```
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\user\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore
  if pd.api.types.is_categorical_dtype(vector):
```



Campaign 4 had the highest acceptance rate, but the latest campaign, Campaign 5, received responses from 15% of the audience, indicating a notable performance in terms of engagement.

Next, let's delve into the income distribution of the people who responded to the campaigns.

```
plt.hist(mkt_data[mkt_data["Response"]==1]["Income"])
plt.show()

sns.boxplot(y=mkt_data[mkt_data["Response"]==1]["Income"])
plt.show()
```