MAVEN MARKETING CAMPAIGN ANALYSIS

Marketing campaign data of 2,240 customers of Maven Marketing, including customer profiles, product preferences, campaign successes/failures, and channel performance.

About the Dataset

Customer Profile

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 customer complained in the last 2 years, 0 otherwise

Country: Customer's location

Product Preferences

MntWines: Amount spent on wine in the last 2 years

MntFruits: Amount spent on fruits in the last 2 years

MntMeatProducts: Amount spent on meat in the last 2 years

MntFishProducts: Amount spent on fish in the last 2 years

MntSweetProducts: Amount spent on sweets in the last 2 years

MntGoldProds: Amount spent on gold in the last 2 years

Channel Performance

NumWebPurchases: Number of purchases made through the company's web site

NumCatalogPurchases: Number of purchases made using a catalogue

NumDealsPurchases: Number of purchases made with a discount

NumStorePurchases: Number of purchases made directly in stores

NumWebVisitsMonth: Number of visits to company's web site in the last month

Campaign Success

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

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving marketing_data.csv to marketing_data.csv

Double-click (or enter) to edit

```
import io

df = pd.read_csv(io.BytesIO(uploaded['marketing_data.csv'])) ## skip initial space present
df
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	D
0	9432	1977	Graduation	Together	$666,666.00	1	0	
1	1503	1976	PhD	Together	$162,397.00	1	1	
2	1501	1982	PhD	Married	$160,803.00	0	0	
3	5336	1971	Master	Together	$157,733.00	1	0	
4	8475	1973	PhD	Married	$157,243.00	0	1	
...	...	...	...	...	...	...	...	
2235	8996	1957	PhD	Married	NaN	2	1	
2236	9235	1957	Graduation	Single	NaN	1	1	
2237	10339	1954	Master	Together	NaN	0	1	
2238	10475	1970	Master	Together	NaN	0	1	
2239	10629	1973	2n Cycle	Married	NaN	1	0	

2240 rows × 28 columns

```
#checking duplicate rows
duplicate = df[df.duplicated()]

print("Duplicate Rows :")

# Print the resultant Dataframe
duplicate
```

```
Duplicate Rows :
```

ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer

0 rows × 28 columns

```
#checkin null values, if any

print(df.isnull().values.any())
```

```
True
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 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   object
 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   object
 10   MntFruits           2240 non-null   object
 11   MntMeatProducts     2240 non-null   object
 12   MntFishProducts     2240 non-null   object
 13   MntSweetProducts    2240 non-null   object
 14   MntGoldProds        2240 non-null   object
 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  Response             2240 non-null   int64 
```

```
 26  Complain             2240 non-null   int64
 27  Country              2240 non-null   object
dtypes: int64(17), object(11)
memory usage: 490.1+ KB
```

```
df.columns
```

```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', '  Income  ',
       'Kidhome', 'Teenhome', 'Dt_Customer', 'Recency', ' MntWines ',
       ' MntFruits ', ' MntMeatProducts ', ' MntFishProducts ',
       ' MntSweetProducts ', ' MntGoldProds ', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
       'AcceptedCmp1', 'AcceptedCmp2', 'Response', 'Complain', 'Country'],
      dtype='object')
```

```
df.describe()
```

	ID	Year_Birth	Kidhome	Teenhome	Recency	NumDealsPurcha
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000
mean	5592.159821	1968.805804	0.444196	0.506250	49.109375	2.325
std	3246.662198	11.984069	0.538398	0.544538	28.962453	1.932
min	0.000000	1893.000000	0.000000	0.000000	0.000000	0.000
25%	2828.250000	1959.000000	0.000000	0.000000	24.000000	1.000
50%	5458.500000	1970.000000	0.000000	0.000000	49.000000	2.000
75%	8427.750000	1977.000000	1.000000	1.000000	74.000000	3.000
max	11191.000000	1996.000000	2.000000	2.000000	99.000000	15.000

Quality issues

1. There are unnecessary spaces in few column names i.e. ' Income ', ' MntWines ', ' MntFruits ', ' MntMeatProducts ', ' MntFishProducts ', ' MntSweetProducts ', ' MntGoldProds '.
2. There are dollar signs, dots, spaces, dashes, zeros after decimal and commas in the values of above mentioned columns, which are errored values. Need to be corrected.
3. Above mentioned columns datatype is object, it should be integer or float as it is representing money.

4. Above mentioned columns have mixed/string datatype (having string and float both), which needs to be corrected.
5. The "Income" column has 23 missing values.
6. Dt_Customer's type is string/object, need to be in datetime format

DATA CLEANING

```
df_clean = df.copy()
```

Removing extra spaces in column names

```
df_clean.rename(columns={'  Income  ':'Income',' MntWines ':'WinesAmount',' MntFruits ':'F
                         ' MntSweetProducts ':'SweetAmount' ,' MntGoldProds ':'GoldAmount'
df_clean.columns
```

```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'WinesAmount', 'FruitsAmount',
       'MeatAmount', 'FishAmount', 'SweetAmount', 'GoldAmount',
       'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',
       'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
       'Response', 'Complain', 'Country'],
      dtype='object')
```

There are dollar signs, spaces, commas, and dots is the values of Income and all amounts column as mentioned

```
df_clean.Income = df_clean.Income.str.replace(".", "")
df_clean.Income = df_clean.Income.str.replace(",", "")
df_clean.Income = df_clean.Income.str.replace("00 ", "")
df_clean.Income = df_clean.Income.str.replace("$", "")
```

```
df_clean.WinesAmount = df_clean.WinesAmount.str.replace(".", "")
df_clean.WinesAmount= df_clean.WinesAmount.str.replace(",", "")
df_clean.WinesAmount = df_clean.WinesAmount.str.replace("00 ", "")
df_clean.WinesAmount = df_clean.WinesAmount.str.replace("$", "")
df_clean.WinesAmount = df_clean.WinesAmount.str.replace("-", "")
```

```
df_clean.FruitsAmount = df_clean.FruitsAmount.str.replace(".", "")
df_clean.FruitsAmount = df_clean.FruitsAmount.str.replace(",", "")
df_clean.FruitsAmount = df_clean.FruitsAmount.str.replace("00 ", "")
df_clean.FruitsAmount = df_clean.FruitsAmount.str.replace("$", "")
```

```
df_clean.FruitsAmount = df_clean.FruitsAmount.str.replace("-", "")


df_clean.MeatAmount = df_clean.MeatAmount.str.replace(".", "")
df_clean.MeatAmount = df_clean.MeatAmount.str.replace(",", "")
df_clean.MeatAmount = df_clean.MeatAmount.str.replace("00 ", "")
df_clean.MeatAmount = df_clean.MeatAmount.str.replace("$", "")
df_clean.MeatAmount = df_clean.MeatAmount.str.replace("-", "")


df_clean.FishAmount = df_clean.FishAmount.str.replace(".", "")
df_clean.FishAmount = df_clean.FishAmount.str.replace(",", "")
df_clean.FishAmount = df_clean.FishAmount.str.replace("00 ", "")
df_clean.FishAmount = df_clean.FishAmount.str.replace("$", "")
df_clean.FishAmount = df_clean.FishAmount.str.replace("-", "")


df_clean.GoldAmount = df_clean.GoldAmount.str.replace(".", "")
df_clean.GoldAmount = df_clean.GoldAmount.str.replace(",", "")
df_clean.GoldAmount = df_clean.GoldAmount.str.replace("00 ", "")
df_clean.GoldAmount = df_clean.GoldAmount.str.replace("$", "")
df_clean.GoldAmount = df_clean.GoldAmount.str.replace("-", "")


df_clean.SweetAmount = df_clean.SweetAmount.str.replace("00 ", "")
df_clean.SweetAmount = df_clean.SweetAmount.str.replace(",", "")
df_clean.SweetAmount = df_clean.SweetAmount.str.replace(".", "")
df_clean.SweetAmount = df_clean.SweetAmount.str.replace("$", "")
df_clean.SweetAmount = df_clean.SweetAmount.str.replace("-", "")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: The de
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning: The de
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: FutureWarning: The de
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: FutureWarning: The d
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: FutureWarning: The d

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:21: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:28: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:31: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:35: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:38: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: FutureWarning: The d
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: FutureWarning: The d
```

```
#checking whether errored values are removed or not
example = df_clean[['Income', 'WinesAmount', 'FruitsAmount']]
example
```

	Income	WinesAmount	FruitsAmount
0	666666	9	14
1	162397	85	1
2	160803	55	16
3	157733	39	1
4	157243	20	2
...	...	...	...
2235	NaN	230	42
2236	NaN	7	
2237	NaN	161	
2238	NaN	187	5
2239	NaN	25	3

2240 rows × 3 columns

Treating missing values in the Income column

```
df_clean.Income.tail(26)
```

```
2214     2447
2215     1730
2216      NaN
2217      NaN
2218      NaN
2219      NaN
2220      NaN
2221      NaN
2222      NaN
2223      NaN
2224      NaN
2225      NaN
2226      NaN
2227      NaN
2228      NaN
2229      NaN
2230      NaN
2231      NaN
2232      NaN
2233      NaN
2234      NaN
2235      NaN
2236      NaN
2237      NaN
2238      NaN
2239      NaN
Name: Income, dtype: object
```

```
df_clean['Income'].apply(type).value_counts()    #there are 24 NaN values, which are bein
```

```
<class 'str'>      2216
<class 'float'>      24
Name: Income, dtype: int64
```

```
df_clean['FruitsAmount'].apply(type).value_counts()
```

```
<class 'str'>    2240
Name: FruitsAmount, dtype: int64
```

```
df_clean['GoldAmount'].apply(type).value_counts()
```

```
<class 'str'>    2240
Name: GoldAmount, dtype: int64
```

```
df_clean['SweetAmount'].apply(type).value_counts()
```

```
<class 'str'>    2240
Name: SweetAmount, dtype: int64
```

```
df_clean['WinesAmount'].apply(type).value_counts()
```

```
<class 'str'>    2240
Name: WinesAmount, dtype: int64
```

```
df_clean['MeatAmount'].apply(type).value_counts()
```

```
<class 'str'>    2240
Name: MeatAmount, dtype: int64
```

```
#after removing the errored values, some values in the Amounts column have become empty st
df_clean.FruitsAmount.head(10)
```

```
0      14
1       1
2      16
3       1
4       2
5
6       1
7       1
8       2
9     181
Name: FruitsAmount, dtype: object
```

```
pd.to_numeric(df_clean['FruitsAmount'], errors='coerce')
```

```
0       14.0
1        1.0
2       16.0
3        1.0
4        2.0
        ...
2235    42.0
2236     NaN
2237     NaN
2238     5.0
2239     3.0
Name: FruitsAmount, Length: 2240, dtype: float64
```

```
#replacing empty strings with NaN value
import numpy as np
df_clean = df_clean.replace(r'^\s*$', np.nan, regex=True)   #pandas replace empty string w
```

```
df_clean
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	9432	1977	Graduation	Together	666666	1	0	02/0
1	1503	1976	PhD	Together	162397	1	1	03/0
2	1501	1982	PhD	Married	160803	0	0	04/0
3	5336	1971	Master	Together	157733	1	0	04/0
4	8475	1973	PhD	Married	157243	0	1	01/0
...	...	...	...	...	...	...	...	
2235	8996	1957	PhD	Married	NaN	2	1	19/1
2236	9235	1957	Graduation	Single	NaN	1	1	27/0
2237	10339	1954	Master	Together	NaN	0	1	23/0
2238	10475	1970	Master	Together	NaN	0	1	01/0
2239	10629	1973	2n Cycle	Married	NaN	1	0	14/0

2240 rows × 28 columns

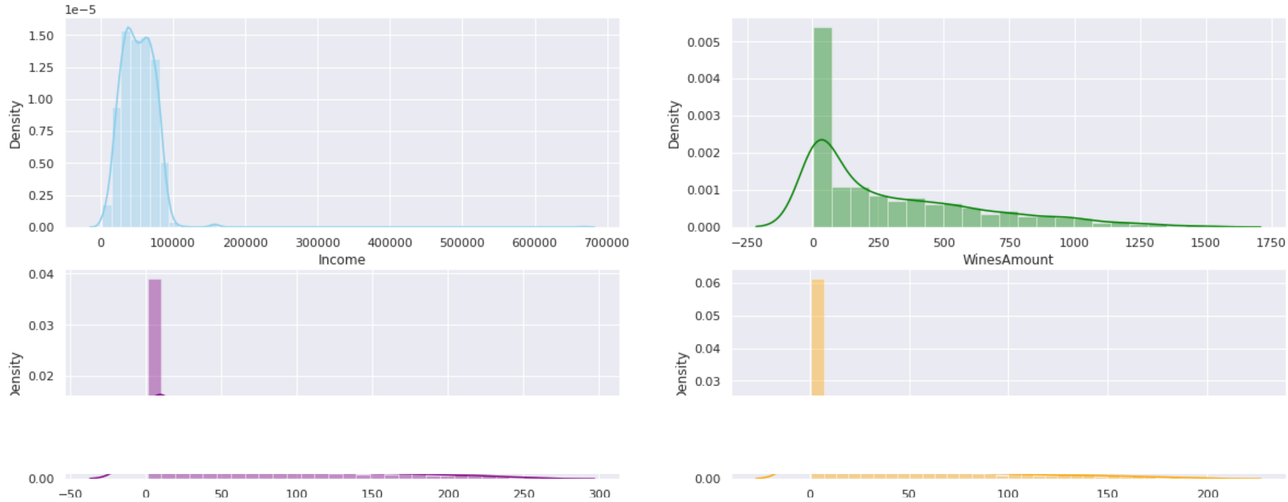
Dealing with missing or NaN values of the dataframe

1. First, plotting the features with NaN values to understand, which imputation should be done. i.e. mean,media,mode

2. When the data is skewed, it is good to consider using the median value for replacing the missing values
3. For symmetric data distribution, one can use the mean value for imputing missing values.

```
#libraries & dataset

import seaborn as sns
import matplotlib.pyplot as plt

# set a grey background (use sns.set_theme() if seaborn version 0.11.0 or above)
sns.set(style="darkgrid")

fig, axs = plt.subplots(3, 2, figsize=(20,12))

sns.distplot(df_clean.Income, kde= True, color="skyblue", ax=axs[0, 0])
sns.distplot(df_clean.WinesAmount, kde=True, color="green", ax=axs[0, 1])
sns.distplot(df_clean.FishAmount, kde=True, color="purple", ax=axs[1, 0])
sns.distplot(df_clean.FruitsAmount, kde=True, color="orange", ax=axs[1, 1 ])
sns.distplot(df_clean.MeatAmount,  kde=True, color="maroon", ax=axs[2, 0])
sns.distplot(df_clean.GoldAmount,  kde=True, color="yellow", ax=axs[2, 1])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f4239ad0>
```



Fill NaN Values in All Columns with MEDIAN value of each column

*ALL THE PLOTS ARE POSITIVELY SKEWED, THUS **MEDIAN IMPUTATION** SHOULD BE DONE*

```
df_new = df_clean.fillna(df_clean.median())
df_new
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Droppi
  """Entry point for launching an IPython kernel.
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cu
0	9432	1977	Graduation	Together	666666	1	0	02/(
1	1503	1976	PhD	Together	162397	1	1	03/(

```
#now checking if there is any null value in  the new dataframe

print(df_new.isnull().values.any())
```

```
False
```

Fixing Data types of Income and Amount columns

```
df_new.dtypes
```

```
ID                    int64
Year_Birth            int64
Education            object
Marital_Status       object
Income               object
Kidhome               int64
Teenhome              int64
Dt_Customer          object
Recency               int64
WinesAmount          object
FruitsAmount         object
MeatAmount           object
FishAmount           object
SweetAmount          object
GoldAmount           object
NumDealsPurchases     int64
NumWebPurchases       int64
NumCatalogPurchases   int64
NumStorePurchases     int64
NumWebVisitsMonth     int64
AcceptedCmp3          int64
AcceptedCmp4          int64
AcceptedCmp5          int64
AcceptedCmp1          int64
AcceptedCmp2          int64
Response              int64
Complain              int64
Country              object
dtype: object
```

```
df_new['WinesAmount'].apply(type).value_counts()
```

```
<class 'str'>      2227
<class 'float'>      13
Name: WinesAmount, dtype: int64
```

```
df_new['MeatAmount'].apply(type).value_counts()
```

```
<class 'str'>      2239
<class 'float'>       1
Name: MeatAmount, dtype: int64
```

```
df_new['GoldAmount'].apply(type).value_counts()
```

```
<class 'str'>      2179
<class 'float'>      61
Name: GoldAmount, dtype: int64
```

```
df_new['Income'].apply(type).value_counts()
```

```
<class 'str'>      2216
<class 'float'>      24
Name: Income, dtype: int64
```

There are mixed values i.e. string/floats in the above columns, they need to be converted into integer values.

```
df_new['In_Type'] = df_new['FishAmount'].apply(lambda x: type(x).__name__)
```

```
df_new.In_Type.tail(25)
```

```
2215      str
2216      str
2217    float
2218    float
2219      str
2220      str
2221      str
2222      str
2223    float
2224      str
2225    float
2226      str
2227      str
2228      str
2229      str
2230      str
2231      str
2232      str
2233      str
2234      str
2235      str
2236      str
2237    float
2238      str
2239      str
Name: In_Type, dtype: object
```

```
df_new.Income = df_new.Income.astype(int)

df_new.WinesAmount = df_new.WinesAmount.astype(int)
df_new.FruitsAmount = df_new.FruitsAmount.astype(int)
df_new.FishAmount = df_new.FishAmount.astype(int)
df_new.GoldAmount = df_new.GoldAmount.astype(int)
df_new.MeatAmount = df_new.MeatAmount.astype(int)
df_new.SweetAmount = df_new.SweetAmount.astype(int)

df_new.Dt_Customer = pd.to_datetime(df_new.Dt_Customer)


df_new.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               2240 non-null   int64         
 5   Kidhome              2240 non-null   int64         
 6   Teenhome             2240 non-null   int64         
 7   Dt_Customer          2240 non-null   datetime64[ns]
 8   Recency              2240 non-null   int64         
 9   WinesAmount          2240 non-null   int64         
 10  FruitsAmount         2240 non-null   int64         
 11  MeatAmount           2240 non-null   int64         
 12  FishAmount           2240 non-null   int64         
 13  SweetAmount          2240 non-null   int64         
 14  GoldAmount           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  Response             2240 non-null   int64         
 26  Complain             2240 non-null   int64         
 27  Country              2240 non-null   object        
 28  In_Type              2240 non-null   object        
dtypes: datetime64[ns](1), int64(24), object(4)
memory usage: 507.6+ KB
```

```
#plotting columns after imputation

import seaborn as sns
import matplotlib.pyplot as plt

# set a grey background (use sns.set_theme() if seaborn version 0.11.0 or above)
```

```
sns.set(style="darkgrid")

fig, axs = plt.subplots(3, 2, figsize=(20,12))

sns.distplot(df_new.Income, kde= True, color="skyblue", ax=axs[0, 0])
sns.distplot(df_new.WinesAmount, kde=True, color="green", ax=axs[0, 1])
sns.distplot(df_new.FishAmount, kde=True, color="purple", ax=axs[1, 0])
sns.distplot(df_new.FruitsAmount, kde=True, color="orange", ax=axs[1, 1 ])
sns.distplot(df_new.MeatAmount,  kde=True, color="maroon", ax=axs[2, 0])
sns.distplot(df_new.GoldAmount,  kde=True, color="yellow", ax=axs[2, 1])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
```

PERFORMING EDA TO IDENTIFY UNDERLYING PATTERNS IN THE DATA

```
  warnings.warn(msg, FutureWarning)
```

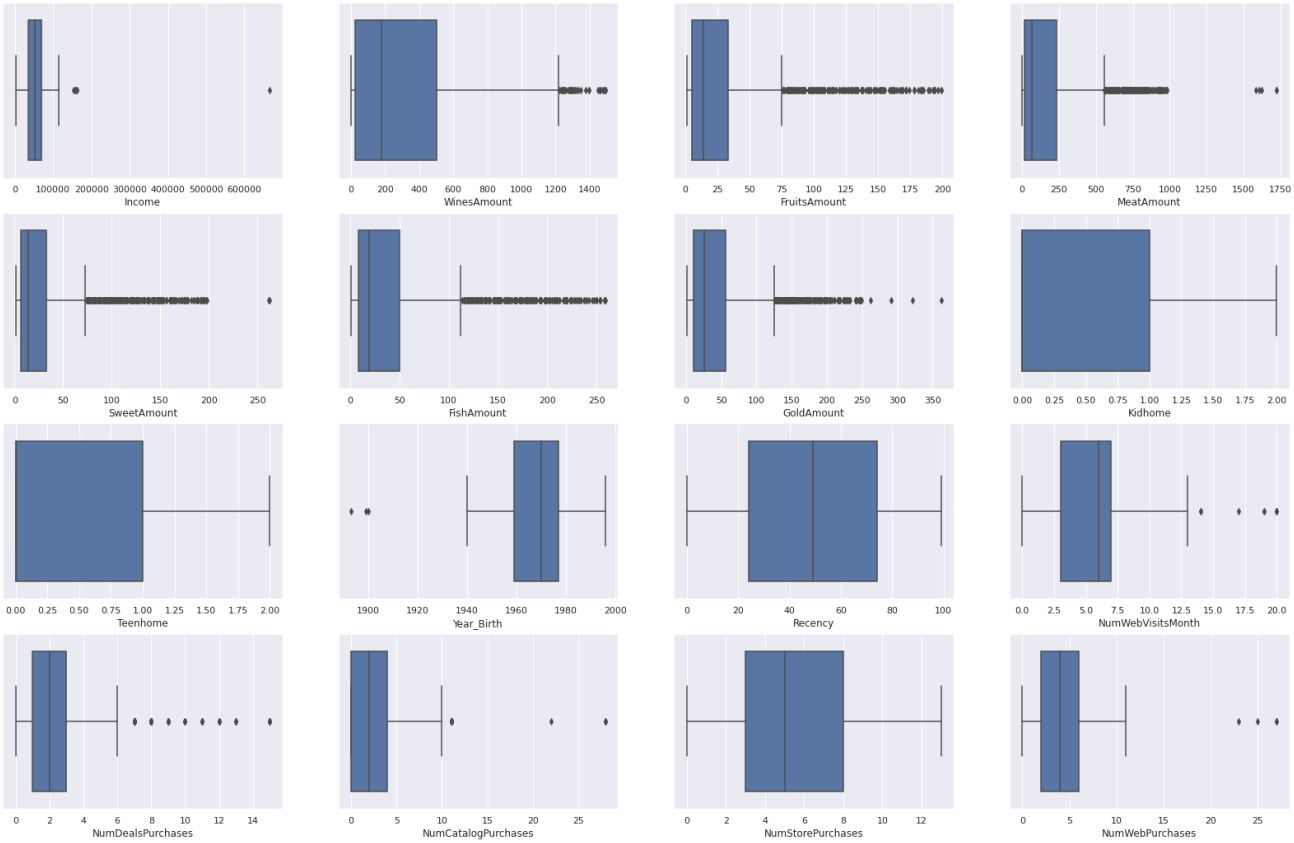
1. CHECKING IF THERE IS ANY OUTLIERS

There are three causes for OUTLIERS:

1. Data entry or measurement errors ---> Trim the data set, Set your range for what's valid
2. Sampling problems and unusual conditions ---> iF, UNABLE TO DECIDE WHETHER WE SHOULD REMOVE IT OR NOT THEN, Trim the data set, but replace outliers with the nearest "good" data, as opposed to truncating them completely. (This called Winsorization.) OR Replace outliers with the mean or median
3. Natural variation --> you should not remove it.


```
# set a grey background (use sns.set_theme() if seaborn version 0.11.0 or above)
sns.set(style="darkgrid")

fig, axs = plt.subplots(4,4, figsize=(28,18))

#PLOTTING THOSE COLUMNS IN WHICH OUTLIERS COULD OCCUR

sns.boxplot(data= df_new, x="Income", ax=axs[0, 0])
sns.boxplot(data= df_new, x="WinesAmount", ax=axs[0,1 ])
sns.boxplot(data= df_new, x="FruitsAmount", ax=axs[0, 2])
sns.boxplot(data= df_new, x="MeatAmount", ax=axs[0, 3])
sns.boxplot(data= df_new, x="SweetAmount", ax=axs[1, 0])
sns.boxplot(data= df_new, x="FishAmount", ax=axs[1, 1])
sns.boxplot(data= df_new, x="GoldAmount", ax=axs[1, 2])
sns.boxplot(data= df_new, x="Kidhome", ax=axs[1, 3])
sns.boxplot(data= df_new, x="Teenhome", ax=axs[2, 0])
sns.boxplot(data= df_new, x="Year_Birth", ax=axs[2, 1])
sns.boxplot(data= df_new, x="Recency", ax=axs[2, 2])
sns.boxplot(data= df_new, x="NumWebVisitsMonth", ax=axs[2, 3])
sns.boxplot(data= df_new, x="NumDealsPurchases", ax=axs[3, 0])
sns.boxplot(data= df_new, x="NumCatalogPurchases", ax=axs[3, 1])
sns.boxplot(data= df_new, x="NumStorePurchases", ax=axs[3, 2])
sns.boxplot(data= df_new, x="NumWebPurchases", ax=axs[3, 3])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f1818b50>
```

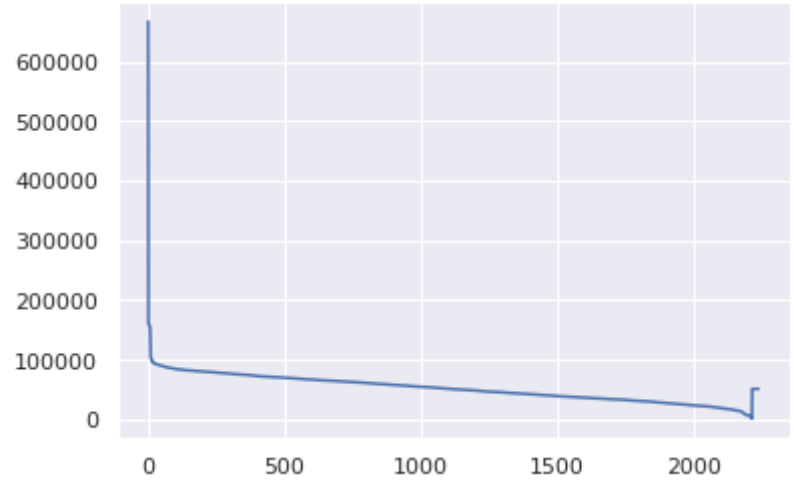


1. 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth' are showing Natural variation as there could be more or less purchase by the customers.
2. There is no outlier in 'Recency', 'kidhome' and 'teenhome'.
3. the outliers in Year_birth seems like entry errors since it's impossible that people who was born before 1900 still alive and participating in marketing campaign. Therefore,the outliers in Year_birth need to be studied. (Reference: https://statisticsbyjim.com/basics/remove-outliers/)

4. 'Income column has also a outlier, not sure should i remove it or not, it could occured naturally or a typinf error, need to study further that outlier data point.
5. 'WinesAmount', 'FruitsAmount', 'MeatAmount', 'FishAmount', 'SweetAmount', 'GoldAmount' has outliers but occuring naturally, because onne can spend more or less on anything.

Figure outing the Income column's outlier

```
plt.plot(df_new.Income)  # Plot the chart
plt.show()
```




```
df_new.Income.describe()  #checking the maximum value of income column, which is the outli
```

```
count      2240.000000
mean      52237.970089
std       25037.956074
min        1730.000000
25%       35538.750000
50%       51381.000000
75%       68289.750000
max      666666.000000
Name: Income, dtype: float64
```

```
df_new[df_new.Income == 666666]
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custome
0	9432	1977	Graduation	Together	666666	1	0	2013-02-(

1 rows × 29 columns

it is not possible that a person with only a graduate degree can earn around 6lacs, so considering it is a typing mistake and replacing it with mean value is suitable in this case as there is only 1 outlier.

```
df_new["Income"] = np.where(df_new["Income"] == 666666 , df_new.Income.median(), df_new["I
```

```
df_new.Income.describe()
```

```
count      2240.000000
mean      51963.289286
std       21405.893964
min        1730.000000
25%       35538.750000
50%       51381.000000
75%       68275.750000
max      162397.000000
Name: Income, dtype: float64
```

```
#plottin after replacing oultier

fig= plt.subplots(figsize=(10,8))
sns.boxplot(data= df_new, x="Income")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f1448950>
```

```
print('skewness value of income: ',df_new['Income'].skew())
```

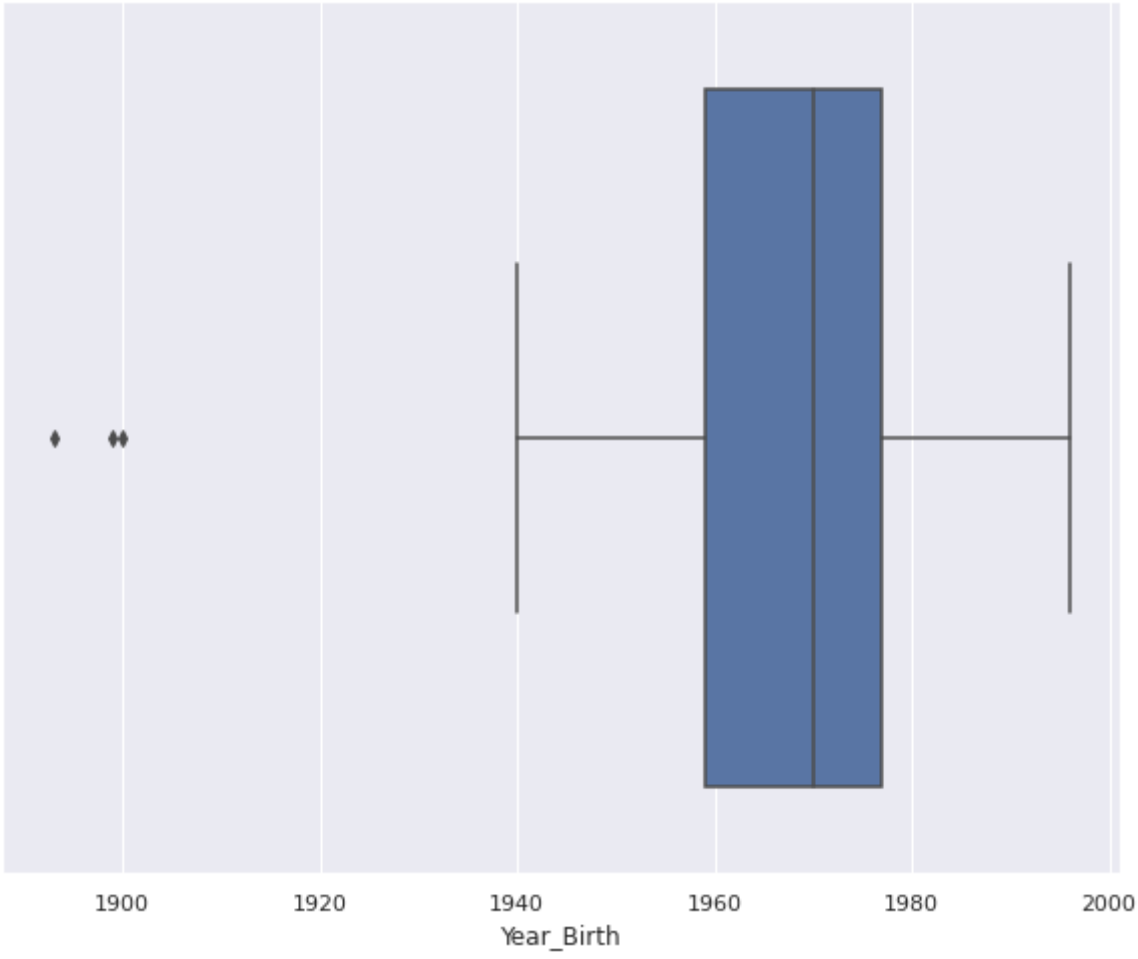
```
skewness value of income:  0.35021902895567913
```

the skewness value should be within the range of -1 to 1 for a normal distribution, any major changes from this value may indicate the presence of outliers.

Figure outing the Year_Birth column's outlier

```
fig= plt.subplots(figsize=(10,8))
sns.boxplot(data= df_new, x="Year_Birth")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f140dc10>
```



the outliers in Year_birth seems like entry errors since it's impossible that people who was born in or before 1900 still alive and participating in marketing campaign.

There are 2 rules for dealing with outliers:

1. Use empirical relations of Normal distribution --- Normal distribution
2. Inter-Quartile Range (IQR) proximity rule ---- skewed distribution

It is being observed that the plot above is a bit skewed , but its better to check, the skewness value should be within the range of -1 to 1 for a normal distribution.

```
print('skewness value of income: ',df_new['Year_Birth'].skew())
```

```
skewness value of income:  -0.34994385918269555
```

Above plot is normaly distributed, therefore using empirical relations of Normal distribution.

The data points which fall below mean-3*(sigma) or above mean+3*(sigma) are outliers and would be changed by upper and lower values.

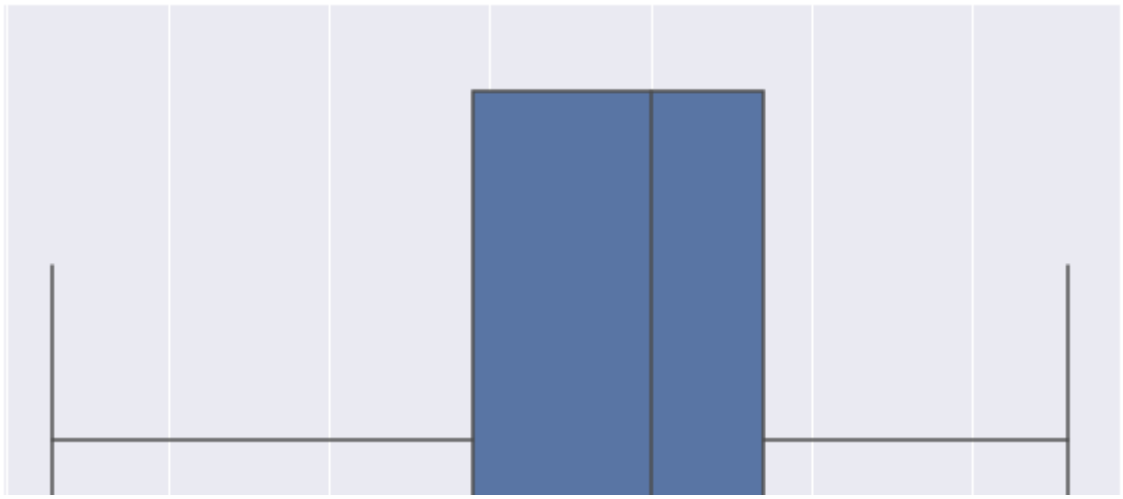
Capping

In this technique, we cap our outliers data and make the limit i.e, above a particular value or less than that value, all the values will be considered as outliers, and the number of outliers in the dataset gives that capping number.

```
#The code below drops the outliers by removing all the values that are below all below mea
```

```
Y_mean = df_new['Year_Birth'].mean()
Y_std = df_new['Year_Birth'].std()
low= Y_mean -(3 * Y_std)
high= Y_mean + (3 * Y_std)

df_new['Year_Birth']= np.where(df_new['Year_Birth']>high , high,
                      np.where(df_new['Year_Birth']<low , low,
                      df_new['Year_Birth']))


fig= plt.subplots(figsize=(10,8))
sns.boxplot(data= df_new, x="Year_Birth")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f18f5cd0>
```



```
df_new.Year_Birth.describe()
```

```
count    2240.000000
mean     1968.853375
std        11.768185
min      1932.853595
25%      1959.000000
50%      1970.000000
75%      1977.000000
max      1996.000000
Name: Year_Birth, dtype: float64
```

```
print('skewness value of income: ',df_new['Year_Birth'].skew())
```

```
skewness value of income:  -0.11780345838511287
```

```
df_new = df_new.drop(columns=['In_Type'])
```

```
#datatype of Income and year column converted to float due to capping, therefore changin i

df_new.Income = df_new.Income.astype(int)
df_new.Year_Birth = df_new.Year_Birth.astype(int)
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 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               2240 non-null   int64         
 5   Kidhome              2240 non-null   int64         
 6   Teenhome             2240 non-null   int64         
 7   Dt_Customer          2240 non-null   datetime64[ns]
 8   Recency              2240 non-null   int64         
```

```
 9   WinesAmount          2240 non-null   int64
 10  FruitsAmount         2240 non-null   int64
 11  MeatAmount           2240 non-null   int64
 12  FishAmount           2240 non-null   int64
 13  SweetAmount          2240 non-null   int64
 14  GoldAmount           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  Response             2240 non-null   int64
 26  Complain             2240 non-null   int64
 27  Country              2240 non-null   object
dtypes: datetime64[ns](1), int64(24), object(3)
memory usage: 490.1+ KB
```

All issues are now fixed

1. No duplicate values
2. No null/missing/NaN values
3. No datatype inconsistencies
4. No invalid data
5. No outliers
6. All checks done

```
df_new
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	9432	1977	Graduation	Together	51381	1	0	2013

```
# store the file
df_new.reset_index(drop=True)
df_new.to_csv('df_new.csv', index=False)
#load data
df = pd.read_csv('df_new.csv')
df
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	9432	1977	Graduation	Together	51381	1	0	2013
1	1503	1976	PhD	Together	162397	1	1	2013
2	1501	1982	PhD	Married	160803	0	0	2012
3	5336	1971	Master	Together	157733	1	0	2013
4	8475	1973	PhD	Married	157243	0	1	2014
...	...	...	...	...	...	...	...	
2235	8996	1957	PhD	Married	51381	2	1	2012
2236	9235	1957	Graduation	Single	51381	1	1	2014
2237	10339	1954	Master	Together	51381	0	1	2013
2238	10475	1970	Master	Together	51381	0	1	2013
2239	10629	1973	2n Cycle	Married	51381	1	0	2012

2240 rows × 28 columns

Data transformation for analysis

```
df.Dt_Customer = pd.to_datetime(df.Dt_Customer)
df.rename(columns={'Response':'LastCampaign'}, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 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               2240 non-null   int64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   datetime64[ns]
 8   Recency              2240 non-null   int64
 9   WinesAmount          2240 non-null   int64
 10  FruitsAmount         2240 non-null   int64
 11  MeatAmount           2240 non-null   int64
 12  FishAmount           2240 non-null   int64
 13  SweetAmount          2240 non-null   int64
 14  GoldAmount           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  LastCampaign         2240 non-null   int64
 26  Complain             2240 non-null   int64
 27  Country              2240 non-null   object
dtypes: datetime64[ns](1), int64(24), object(3)
memory usage: 490.1+ KB
```

```
df["Join_year"] = df.Dt_Customer.dt.year
df["Join_month"] = df.Dt_Customer.dt.month
df["Join_weekday"] = df.Dt_Customer.dt.weekday
df['Age'] =  df.Join_year - df.Year_Birth
df["Children"] = df.Kidhome + df.Teenhome
df['Total_Amount'] = df.WinesAmount+ df.FruitsAmount+ df.MeatAmount+ df.FishAmount+ df.Swe
df['Total_purchase_Count'] = df.NumDealsPurchases+ df.NumWebPurchases+df.NumCatalogPurchas
df['Total_accepted_campaign'] = df.AcceptedCmp1 + df.AcceptedCmp2 + df.AcceptedCmp3 + df.A
df['Average_PurchaseAmount'] = df.Total_Amount/df.Total_purchase_Count
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 37 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                   2240 non-null   int64
 5   Kidhome                  2240 non-null   int64
 6   Teenhome                 2240 non-null   int64
 7   Dt_Customer              2240 non-null   datetime64[ns]
 8   Recency                  2240 non-null   int64
 9   WinesAmount              2240 non-null   int64
```

```
 10  FruitsAmount             2240 non-null   int64
 11  MeatAmount               2240 non-null   int64
 12  FishAmount               2240 non-null   int64
 13  SweetAmount              2240 non-null   int64
 14  GoldAmount               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  LastCampaign             2240 non-null   int64
 26  Complain                 2240 non-null   int64
 27  Country                  2240 non-null   object
 28  Join_year                2240 non-null   int64
 29  Join_month               2240 non-null   int64
 30  Join_weekday             2240 non-null   int64
 31  Age                      2240 non-null   int64
 32  Children                 2240 non-null   int64
 33  Total_Amount             2240 non-null   int64
 34  Total_purchase_Count     2240 non-null   int64
 35  Total_accepted_campaign  2240 non-null   int64
 36  Average_PurchaseAmount   2240 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(32), object(3)
memory usage: 647.6+ KB
```

After indepth evaluation of Average_PurchaseAmount column, it is being noticed that there are some 'inf' values which came out be infinity because of zero value of Total_purchase_Count.

```
df.Average_PurchaseAmount.describe()
```

```
count    2240.000000
mean             inf
std              NaN
min         0.533333
25%        13.267045
50%        24.343301
75%        46.633929
max              inf
Name: Average_PurchaseAmount, dtype: float64
```

```
df.sort_values(by=['Average_PurchaseAmount'])   #sorting out the Average_PurchaseAmount co
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
2215	6862	1971	Graduation	Divorced	1730	0	0	2014
2212	9931	1963	PhD	Married	4023	1	1	2014
2118	6528	1982	Master	Together	18492	1	0	2014
2126	6742	1979	Graduation	Married	17688	1	0	2013
1993	10914	1970	Graduation	Single	24163	1	1	2013
...	...	...	...	...	...	...	...	
2234	8720	1978	2n Cycle	Together	51381	0	0	2012
2213	11110	1973	Graduation	Single	3502	1	0	2013
6	11181	1949	PhD	Married	156924	0	0	2013
2210	3955	1965	Graduation	Divorced	4861	0	0	2014

```
#max value for Average_PurchaseAmount is 1679, we have to figure out records which are rea
df[df.Average_PurchaseAmount > 1679]
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
6	11181	1949	PhD	Married	156924	0	0	2013
7	5555	1975	Graduation	Divorced	153924	0	0	2014
2210	3955	1965	Graduation	Divorced	4861	0	0	2014
2213	11110	1973	Graduation	Single	3502	1	0	2013

4 rows × 37 columns

It is not possible that a customer has bought or spent his money on Wines,Fruit,Meat etc but has a purchase count np.zeros

There must be some error may be data entry error or customer might forgot to fil out the 'NumDealsPurchases, NumWebPurchases, NumCatalogPurchases,NumStorePurchases' column, Thus, we cannot move forward with our analysis with these errornous records.

Removing these records would be suitable in this scenario, as we dont know their purchase count.

```
# delete a few specified rows at index values 6,7,2210,2213, which has Average_PurchaseAmo
# Note that the index values do not always align to row numbers.
df = df.drop(labels=[ 6,7,2210,2213], axis=0)
```

```
df
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	9432	1977	Graduation	Together	51381	1	0	2013
1	1503	1976	PhD	Together	162397	1	1	2013
2	1501	1982	PhD	Married	160803	0	0	2012
3	5336	1971	Master	Together	157733	1	0	2013
4	8475	1973	PhD	Married	157243	0	1	2014
...	...	...	...	...	...	...	...	
2235	8996	1957	PhD	Married	51381	2	1	2012
2236	9235	1957	Graduation	Single	51381	1	1	2014
2237	10339	1954	Master	Together	51381	0	1	2013
2238	10475	1970	Master	Together	51381	0	1	2013
2239	10629	1973	2n Cycle	Married	51381	1	0	2012

2236 rows × 37 columns

```
#storing the final dataframe in the desktop for further visualzation

df.to_csv (r'C:\Users\Wisha Raees\Downloads\marketing_clean.csv', index = False, header=Tr


df
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
1	1503	1976	PhD	Together	162397	1	1	2013
2	1501	1982	PhD	Married	160803	0	0	2012
3	5336	1971	Master	Together	157733	1	0	2013
4	8475	1973	PhD	Married	157243	0	1	2014
...	...	...	...	...	...	...	...	
2235	8996	1957	PhD	Married	51381	2	1	2012
2236	9235	1957	Graduation	Single	51381	1	1	2014
2237	10339	1954	Master	Together	51381	0	1	2013
2238	10475	1970	Master	Together	51381	0	1	2013
2239	10629	1973	2n Cycle	Married	51381	1	0	2012

2236 rows × 37 columns