In [2]:
```
# import libraries

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

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

## Loading dataset

In [3]:
```
mar_df = pd.read_csv('marketing_data.csv')
```

In [4]:
```
mar_df
```

Out[4]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | F |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | $84,835.00 | 0 | 0 | 6/16/14 | |
| **1** | 1 | 1961 | Graduation | Single | $57,091.00 | 0 | 0 | 6/15/14 | |
| **2** | 10476 | 1958 | Graduation | Married | $67,267.00 | 0 | 1 | 5/13/14 | |
| **3** | 1386 | 1967 | Graduation | Together | $32,474.00 | 1 | 1 | 5/11/14 | |
| **4** | 5371 | 1989 | Graduation | Single | $21,474.00 | 1 | 0 | 4/8/14 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | $66,476.00 | 0 | 1 | 3/7/13 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | $31,056.00 | 1 | 0 | 1/22/13 | |
| **2237** | 22 | 1976 | Graduation | Divorced | $46,310.00 | 1 | 0 | 12/3/12 | |
| **2238** | 528 | 1978 | Graduation | Married | $65,819.00 | 0 | 0 | 11/29/12 | |
| **2239** | 4070 | 1969 | PhD | Married | $94,871.00 | 0 | 2 | 9/1/12 | |

2240 rows × 28 columns

In [5]:
```
# show all columns
pd.set_option('display.max_columns', None)

mar_df
```

Out[5]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | F |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | $84,835.00 | 0 | 0 | 6/16/14 | |
| **1** | 1 | 1961 | Graduation | Single | $57,091.00 | 0 | 0 | 6/15/14 | |
| **2** | 10476 | 1958 | Graduation | Married | $67,267.00 | 0 | 1 | 5/13/14 | |
| **3** | 1386 | 1967 | Graduation | Together | $32,474.00 | 1 | 1 | 5/11/14 | |
| **4** | 5371 | 1989 | Graduation | Single | $21,474.00 | 1 | 0 | 4/8/14 | |

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | R |
|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | $66,476.00 | 0 | 1 | 3/7/13 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | $31,056.00 | 1 | 0 | 1/22/13 | |
| **2237** | 22 | 1976 | Graduation | Divorced | $46,310.00 | 1 | 0 | 12/3/12 | |
| **2238** | 528 | 1978 | Graduation | Married | $65,819.00 | 0 | 0 | 11/29/12 | |
| **2239** | 4070 | 1969 | PhD | Married | $94,871.00 | 0 | 2 | 9/1/12 | |

2240 rows × 28 columns

## Overview:

**The dataset contains customer information with 28 features related to a marketing campaign conducted by a company**

In [6]:
```
##############################################################################
```

# 1.1 High-level Statistics of the Data Set

## 1) Number of Data-points and Features

In [7]:
```
mar_df.shape
```

Out[7]:
```
(2240, 28)
```

In [8]:
```
mar_df.shape[0], mar_df.shape[1]
```

Out[8]:
```
(2240, 28)
```

In [9]:
```
mar_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   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  Response             2240 non-null   int64
 26  Complain             2240 non-null   int64
 27  Country              2240 non-null   object
dtypes: int64(23), object(5)
memory usage: 490.1+ KB
```

- The dataset contains 2240 customers' and 28 features. Each row represents data corresponding to each individual

In [10]:
```
# 28 features

list(mar_df.columns)
```

Out[10]:
```
['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']
```

In [11]:
```
##############################################################################
```

## 1.2 Objective

- To analyse the dataset and draw useful insights about the marketing campaigns'

In [12]:
```
##############################################################################
```

## Data Cleaning, Manipulation and Feature Engineering

In [13]:
```
mar_df
```

Out[13]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | R |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | $84,835.00 | 0 | 0 | 6/16/14 | |
| **1** | 1 | 1961 | Graduation | Single | $57,091.00 | 0 | 0 | 6/15/14 | |
| **2** | 10476 | 1958 | Graduation | Married | $67,267.00 | 0 | 1 | 5/13/14 | |
| **3** | 1386 | 1967 | Graduation | Together | $32,474.00 | 1 | 1 | 5/11/14 | |
| **4** | 5371 | 1989 | Graduation | Single | $21,474.00 | 1 | 0 | 4/8/14 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | $66,476.00 | 0 | 1 | 3/7/13 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | $31,056.00 | 1 | 0 | 1/22/13 | |
| **2237** | 22 | 1976 | Graduation | Divorced | $46,310.00 | 1 | 0 | 12/3/12 | |
| **2238** | 528 | 1978 | Graduation | Married | $65,819.00 | 0 | 0 | 11/29/12 | |
| **2239** | 4070 | 1969 | PhD | Married | $94,871.00 | 0 | 2 | 9/1/12 | |

2240 rows × 28 columns

In [14]:
```
mar_df.describe()
```

Out[14]:

| | ID | Year_Birth | Kidhome | Teenhome | Recency | MntWines | MntFruits | M |
|---|---|---|---|---|---|---|---|---|
| **count** | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | |
| **mean** | 5592.159821 | 1968.805804 | 0.444196 | 0.506250 | 49.109375 | 303.935714 | 26.302232 | |
| **std** | 3246.662198 | 11.984069 | 0.538398 | 0.544538 | 28.962453 | 336.597393 | 39.773434 | |
| **min** | 0.000000 | 1893.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **25%** | 2828.250000 | 1959.000000 | 0.000000 | 0.000000 | 24.000000 | 23.750000 | 1.000000 | |
| **50%** | 5458.500000 | 1970.000000 | 0.000000 | 0.000000 | 49.000000 | 173.500000 | 8.000000 | |
| **75%** | 8427.750000 | 1977.000000 | 1.000000 | 1.000000 | 74.000000 | 504.250000 | 33.000000 | |
| **max** | 11191.000000 | 1996.000000 | 2.000000 | 2.000000 | 99.000000 | 1493.000000 | 199.000000 | |

- The describe() function gives us the 5-number statistical summary of the dataset (minimum, 25th percentile, 50th percentile(median), 75th percentile, maximum) and also the mean and standard deviation.
- By observing the summary above, there are considerable differences between mean and median of certain features due to the presence of outliers. Also significant differences between 75th percentile and 100th percentile (max value) for some features due to the presence of extreme outliers.

In [15]:
```
mar_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   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  Response             2240 non-null   int64 
 26  Complain             2240 non-null   int64 
 27  Country              2240 non-null   object
dtypes: int64(23), object(5)
memory usage: 490.1+ KB
```

In [16]:
```
mar_df.columns
```

Out[16]:
```
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')
```

- column ' Income ' has extra white spaces and the datatype is an object/string

- removing white spaces
- removing unnecessary characters and then converting the datatype to 'float' as 'Income' should be a number but not a string

In [17]:
```
mar_df.columns = mar_df.columns.str.replace(' ', '')
mar_df.columns
```

Out[17]:
```
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')
```

In [18]:
```
mar_df['Income']
```

Out[18]:
```
0       $84,835.00
1       $57,091.00
2       $67,267.00
3       $32,474.00
4       $21,474.00
           ...
2235    $66,476.00
2236    $31,056.00
2237    $46,310.00
2238    $65,819.00
2239    $94,871.00
Name: Income, Length: 2240, dtype: object
```

In [19]:
```
mar_df['Income'] = mar_df['Income'].str.replace('$', '').str.replace(',', '')
```

In [20]:
```
mar_df['Income']
```

Out[20]:
```
0       84835.00
1       57091.00
2       67267.00
3       32474.00
4       21474.00
          ...
2235    66476.00
2236    31056.00
2237    46310.00
2238    65819.00
2239    94871.00
Name: Income, Length: 2240, dtype: object
```

In [21]:
```
mar_df['Income'] = mar_df['Income'].astype('float')
mar_df['Income']
```

Out[21]:
```
0       84835.0
1       57091.0
2       67267.0
3       32474.0
4       21474.0
         ...
2235    66476.0
2236    31056.0
2237    46310.0
2238    65819.0
2239    94871.0
Name: Income, Length: 2240, dtype: float64
```

In [22]:
```
mar_df
```

Out[22]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Rec |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | 84835.0 | 0 | 0 | 6/16/14 | |
| **1** | 1 | 1961 | Graduation | Single | 57091.0 | 0 | 0 | 6/15/14 | |
| **2** | 10476 | 1958 | Graduation | Married | 67267.0 | 0 | 1 | 5/13/14 | |
| **3** | 1386 | 1967 | Graduation | Together | 32474.0 | 1 | 1 | 5/11/14 | |
| **4** | 5371 | 1989 | Graduation | Single | 21474.0 | 1 | 0 | 4/8/14 | |

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Rec |
|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | 66476.0 | 0 | 1 | 3/7/13 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | 31056.0 | 1 | 0 | 1/22/13 | |
| **2237** | 22 | 1976 | Graduation | Divorced | 46310.0 | 1 | 0 | 12/3/12 | |
| **2238** | 528 | 1978 | Graduation | Married | 65819.0 | 0 | 0 | 11/29/12 | |
| **2239** | 4070 | 1969 | PhD | Married | 94871.0 | 0 | 2 | 9/1/12 | |

2240 rows × 28 columns

In [23]:
```
mar_df['Dt_Customer']
```

Out[23]:
```
0        6/16/14
1        6/15/14
2        5/13/14
3        5/11/14
4         4/8/14
          ...
2235      3/7/13
2236     1/22/13
2237     12/3/12
2238    11/29/12
2239      9/1/12
Name: Dt_Customer, Length: 2240, dtype: object
```

- 'Dt_Customer' is stored as a string. Need to convert to datetime

In [24]:
```
mar_df['Dt_Customer'] = pd.to_datetime(mar_df['Dt_Customer'])
mar_df['Dt_Customer']
```

Out[24]:
```
0      2014-06-16
1      2014-06-15
2      2014-05-13
3      2014-05-11
4      2014-04-08
          ...
2235   2013-03-07
2236   2013-01-22
2237   2012-12-03
2238   2012-11-29
2239   2012-09-01
Name: Dt_Customer, Length: 2240, dtype: datetime64[ns]
```

## Handling Missing Values

In [25]:
```
columns = list(mar_df.columns)

print('Missing values for each feature:\n')

for column in columns:
    null = mar_df[column].isnull().sum()
    print(column, ':',  null)
```

```
Missing values for each feature:
```

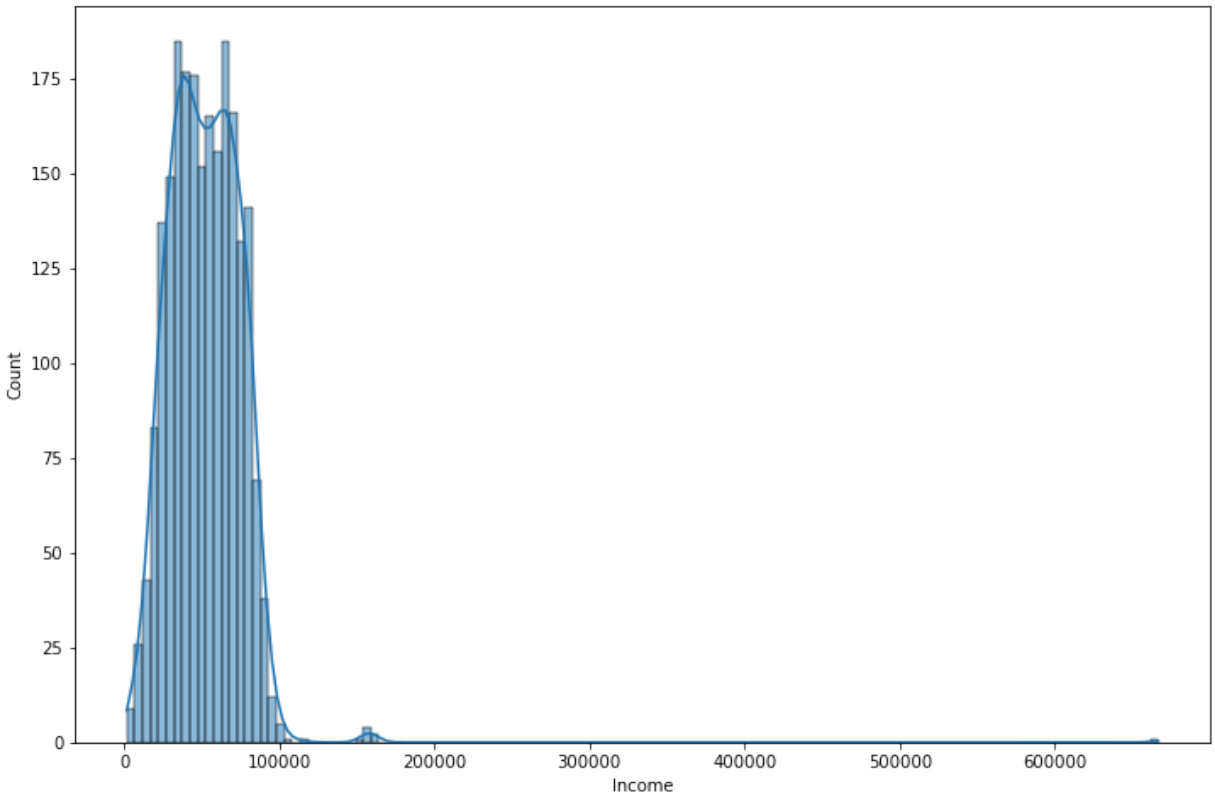
```
ID : 0
Year_Birth : 0
Education : 0
Marital_Status : 0
Income : 24
Kidhome : 0
Teenhome : 0
Dt_Customer : 0
Recency : 0
MntWines : 0
MntFruits : 0
MntMeatProducts : 0
MntFishProducts : 0
MntSweetProducts : 0
MntGoldProds : 0
NumDealsPurchases : 0
NumWebPurchases : 0
NumCatalogPurchases : 0
NumStorePurchases : 0
NumWebVisitsMonth : 0
AcceptedCmp3 : 0
AcceptedCmp4 : 0
AcceptedCmp5 : 0
AcceptedCmp1 : 0
AcceptedCmp2 : 0
Response : 0
Complain : 0
Country : 0
```

- 'Income' feature has 24 missing values
- All other features have no missing values

Plotting the distribution of 'Income'

In [26]:
```
plt.figure(figsize=(12,8))
sns.histplot(data=mar_df['Income'], kde=True)
```

Out[26]:
```
<AxesSubplot:xlabel='Income', ylabel='Count'>
```



- Observation: Majority of the incomes are distributed between $0-100000$ Dollars with few outliers.
- We can impute the missing values with the median to avoid the effect of outliers

In [27]:
```
mar_df['Income'] = mar_df['Income'].fillna(mar_df['Income'].median())
```

In [28]:
```
# checking for null values
mar_df['Income'].isnull().sum()
```

Out[28]:
```
0
```

In [29]:
```
mar_df
```

Out[29]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Rec |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | 84835.0 | 0 | 0 | 2014-06-16 | |
| **1** | 1 | 1961 | Graduation | Single | 57091.0 | 0 | 0 | 2014-06-15 | |
| **2** | 10476 | 1958 | Graduation | Married | 67267.0 | 0 | 1 | 2014-05-13 | |
| **3** | 1386 | 1967 | Graduation | Together | 32474.0 | 1 | 1 | 2014-05-11 | |
| **4** | 5371 | 1989 | Graduation | Single | 21474.0 | 1 | 0 | 2014-04-08 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | 66476.0 | 0 | 1 | 2013-03-07 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | 31056.0 | 1 | 0 | 2013-01-22 | |
| **2237** | 22 | 1976 | Graduation | Divorced | 46310.0 | 1 | 0 | 2012-12-03 | |
| **2238** | 528 | 1978 | Graduation | Married | 65819.0 | 0 | 0 | 2012-11-29 | |
| **2239** | 4070 | 1969 | PhD | Married | 94871.0 | 0 | 2 | 2012-09-01 | |

2240 rows × 28 columns

In [30]:
```
mar_df.columns
```

Out[30]:
```
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')
```

## Feature Engineering

We can engineer new features from the available features in the raw data. This would help us in better understanding of the raw data and for better analysis.

- The total number of children can be engineered from the features 'Kidhome' and 'Teenhome'

  `'Kidhome' + 'Teenhome' = 'Children'`
- The year the person became a customer can be engineered from 'Dt_Customer'
- The total amount spent by each customer can be engineered from sum of all the amounts spent on individual products/items

  `'MntWines' + 'MntFruits' + 'MntMeatProducts' + 'MntFishProducts' + 'MntSweetProducts' + 'MntGoldProds' = 'TotalMntSpent'`
- Total number of deals purchased can be obtained from sum of all the purchases made

  `'NumDealsPurchases' + 'NumWebPurchases' + 'NumCatalogPurchases' + 'NumStorePurchases' = 'Total Purchases'`
- Total number of campaigns accepted by a customer can be obtained from the sum of campaigns accepted

  `'AcceptedCmp3' + 'AcceptedCmp4' + 'AcceptedCmp5' + 'AcceptedCmp1' + 'AcceptedCmp2' + 'Response' = 'TotalCmpAccepted'`

In [31]:
```
mar_df['Children'] = mar_df['Kidhome'] + mar_df['Teenhome']
```

In [32]:
```
mar_df[['Children']]
```

Out[32]:

| | Children |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 1 |
| **3** | 2 |
| **4** | 1 |
| **...** | ... |
| **2235** | 1 |
| **2236** | 1 |
| **2237** | 1 |
| **2238** | 0 |
| **2239** | 2 |

2240 rows × 1 columns

In [33]:
```
mar_df['Year_Customer'] = pd.DatetimeIndex(mar_df['Dt_Customer']).year
```

In [34]:
```
mar_df[['Year_Customer']]
```

Out[34]:

| | Year_Customer |
|---|---|
| **0** | 2014 |
| **1** | 2014 |
| **2** | 2014 |
| **3** | 2014 |
| **4** | 2014 |
| **...** | ... |
| **2235** | 2013 |
| **2236** | 2013 |
| **2237** | 2012 |
| **2238** | 2012 |
| **2239** | 2012 |

2240 rows × 1 columns

In [35]:
```
mar_df['TotalMntSpent'] = mar_df['MntWines'] + mar_df['MntFruits'] + mar_df['MntMeat
```

In [36]:
```
mar_df[['TotalMntSpent']]
```

Out[36]:

| | TotalMntSpent |
|---|---|
| **0** | 1190 |
| **1** | 577 |
| **2** | 251 |
| **3** | 11 |
| **4** | 91 |
| **...** | ... |
| **2235** | 689 |
| **2236** | 55 |
| **2237** | 309 |
| **2238** | 1383 |
| **2239** | 1078 |

2240 rows × 1 columns

In [37]:
```
mar_df
```

Out[37]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Rec |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | 84835.0 | 0 | 0 | 2014-06-16 | |
| **1** | 1 | 1961 | Graduation | Single | 57091.0 | 0 | 0 | 2014-06-15 | |

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Rec |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 10476 | 1958 | Graduation | Married | 67267.0 | 0 | 1 | 2014-05-13 | |
| **3** | 1386 | 1967 | Graduation | Together | 32474.0 | 1 | 1 | 2014-05-11 | |
| **4** | 5371 | 1989 | Graduation | Single | 21474.0 | 1 | 0 | 2014-04-08 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | 66476.0 | 0 | 1 | 2013-03-07 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | 31056.0 | 1 | 0 | 2013-01-22 | |
| **2237** | 22 | 1976 | Graduation | Divorced | 46310.0 | 1 | 0 | 2012-12-03 | |
| **2238** | 528 | 1978 | Graduation | Married | 65819.0 | 0 | 0 | 2012-11-29 | |
| **2239** | 4070 | 1969 | PhD | Married | 94871.0 | 0 | 2 | 2012-09-01 | |

2240 rows × 31 columns

In [38]:
```
mar_df['TotalPurchases'] = mar_df['NumDealsPurchases'] + mar_df['NumWebPurchases'] +
```

In [39]:
```
mar_df[['TotalPurchases']]
```

Out[39]:

| | TotalPurchases |
|---|---|
| **0** | 15 |
| **1** | 18 |
| **2** | 11 |
| **3** | 4 |
| **4** | 8 |
| **...** | ... |
| **2235** | 20 |
| **2236** | 5 |
| **2237** | 14 |
| **2238** | 20 |
| **2239** | 18 |

2240 rows × 1 columns

In [40]:
```
mar_df['TotalCmpAccepted'] = mar_df['AcceptedCmp3'] + mar_df['AcceptedCmp4'] + mar_d
```

In [41]:
```
mar_df[['TotalCmpAccepted']]
```

Out[41]:

| | TotalCmpAccepted |
|---|---|
| **0** | 1 |
| **1** | 2 |

| | TotalCmpAccepted |
|---|---|
| **2** | 0 |
| **3** | 0 |
| **4** | 2 |
| **...** | ... |
| **2235** | 0 |
| **2236** | 0 |
| **2237** | 0 |
| **2238** | 0 |
| **2239** | 3 |

2240 rows × 1 columns

In [42]:
```
mar_df
```

Out[42]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Rec |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | 84835.0 | 0 | 0 | 2014-06-16 | |
| **1** | 1 | 1961 | Graduation | Single | 57091.0 | 0 | 0 | 2014-06-15 | |
| **2** | 10476 | 1958 | Graduation | Married | 67267.0 | 0 | 1 | 2014-05-13 | |
| **3** | 1386 | 1967 | Graduation | Together | 32474.0 | 1 | 1 | 2014-05-11 | |
| **4** | 5371 | 1989 | Graduation | Single | 21474.0 | 1 | 0 | 2014-04-08 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 10142 | 1976 | PhD | Divorced | 66476.0 | 0 | 1 | 2013-03-07 | |
| **2236** | 5263 | 1977 | 2n Cycle | Married | 31056.0 | 1 | 0 | 2013-01-22 | |
| **2237** | 22 | 1976 | Graduation | Divorced | 46310.0 | 1 | 0 | 2012-12-03 | |
| **2238** | 528 | 1978 | Graduation | Married | 65819.0 | 0 | 0 | 2012-11-29 | |
| **2239** | 4070 | 1969 | PhD | Married | 94871.0 | 0 | 2 | 2012-09-01 | |

2240 rows × 33 columns

In [43]:
```
mar_df.drop(columns=['ID', 'Kidhome', 'Teenhome', 'Dt_Customer'], inplace=True)
```

In [44]:
```
mar_df
```

Out[44]:

| | Year_Birth | Education | Marital_Status | Income | Recency | MntWines | MntFruits | MntMeatProdu |
|---|---|---|---|---|---|---|---|---|
| **0** | 1970 | Graduation | Divorced | 84835.0 | 0 | 189 | 104 | |
| **1** | 1961 | Graduation | Single | 57091.0 | 0 | 464 | 5 | |
| **2** | 1958 | Graduation | Married | 67267.0 | 0 | 134 | 11 | |
| **3** | 1967 | Graduation | Together | 32474.0 | 0 | 10 | 0 | |

| | Year_Birth | Education | Marital_Status | Income | Recency | MntWines | MntFruits | MntMeatProdu |
|---|---|---|---|---|---|---|---|---|
| **4** | 1989 | Graduation | Single | 21474.0 | 0 | 6 | 16 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **2235** | 1976 | PhD | Divorced | 66476.0 | 99 | 372 | 18 | 1 |
| **2236** | 1977 | 2n Cycle | Married | 31056.0 | 99 | 5 | 10 | |
| **2237** | 1976 | Graduation | Divorced | 46310.0 | 99 | 185 | 2 | |
| **2238** | 1978 | Graduation | Married | 65819.0 | 99 | 267 | 38 | 7 |
| **2239** | 1969 | PhD | Married | 94871.0 | 99 | 169 | 24 | 5 |

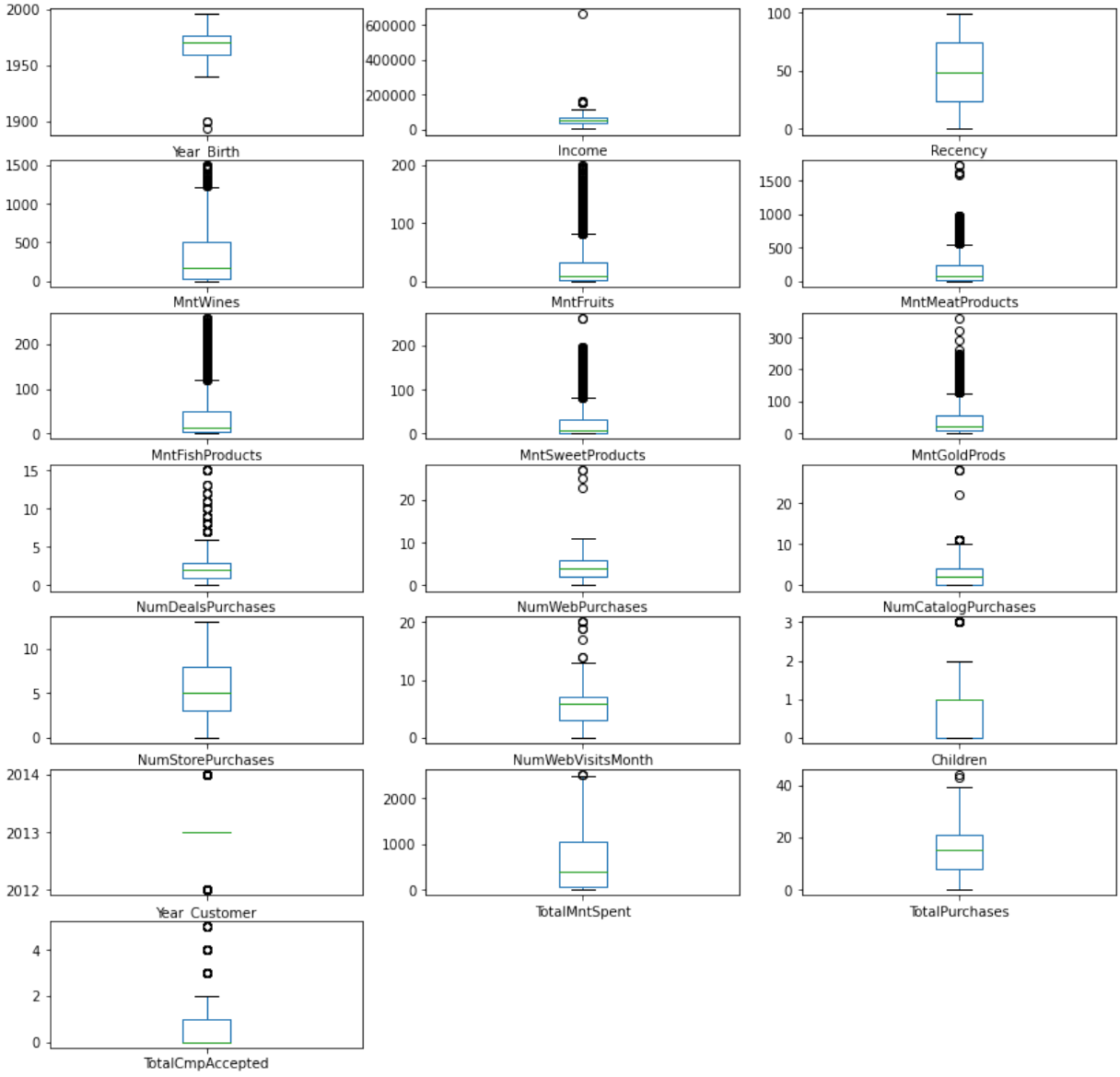
2240 rows × 29 columns

## Detecting and Treating Outliers:

- Box-plot is the best visual plot to detect outliers in the data

In [45]:
```
df_to_plot = mar_df.drop(columns=['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'A
```

In [46]:
```
df_to_plot.plot(subplots=True, layout=(7,3), kind='box', figsize=(14,14))
plt.show()
```




- Here, we will use the concept of percentiles to detect outliers accurately.
- Note: We can also use IQR (Inter Quartile Range) to detect outliers

  `IQR = Q3-Q1 ; where Q1 = 25th percentile value, Q3 = 75th percentile value`

  `Lower Threshold (LT) = Q1 - 1.5 * IQR; all values falling below LT are outliers`
  `Upper Threshold (HT) = Q3 + 1.5 * IQR; all values falling above HT are outliers`
- Extreme outliers:

  `Lower Threshold (LT) = Q1 - 3 * IQR; all values falling below LT are extreme outliers`
  `Upper Threshold (HT) = Q3 + 3 * IQR; all values falling above HT are extreme outliers`

## Detecting outliers using percentiles:

- We will also treat the outliers as per the requirement

In [47]:
```
for i in range(0, 100, 10):
    percentile = mar_df['Year_Birth'].values
```

```
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
0 percentile value is 1893
10 percentile value is 1952
20 percentile value is 1957
30 percentile value is 1962
40 percentile value is 1966
50 percentile value is 1970
60 percentile value is 1973
70 percentile value is 1976
80 percentile value is 1979
90 percentile value is 1984
100 percentile value is  1996
```

In [48]:

```
for i in range(0, 10, 1):
    percentile = mar_df['Year_Birth'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
0 percentile value is 1893
1 percentile value is 1945
2 percentile value is 1947
3 percentile value is 1948
4 percentile value is 1949
5 percentile value is 1950
6 percentile value is 1950
7 percentile value is 1951
8 percentile value is 1951
9 percentile value is 1952
100 percentile value is  1996
```

In [49]:

```
for i in range(90, 100, 1):
    percentile = mar_df['Year_Birth'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 1984
91 percentile value is 1985
92 percentile value is 1986
93 percentile value is 1986
94 percentile value is 1987
95 percentile value is 1988
96 percentile value is 1989
97 percentile value is 1989
98 percentile value is 1990
99 percentile value is 1992
100 percentile value is  1996
```

In [50]:

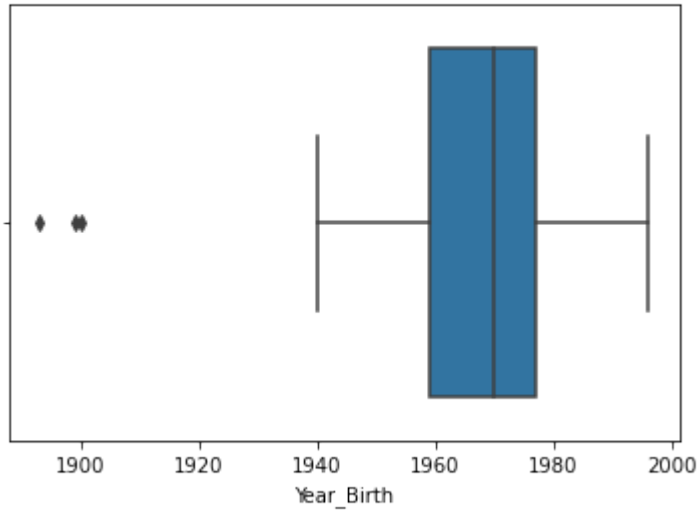
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['Year_Birth'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
# >"1940"
```

```
0.0 percentile value is 1893
0.1 percentile value is 1900
0.2 percentile value is 1941
0.30000000000000004 percentile value is 1943
0.4 percentile value is 1943
0.5 percentile value is 1943
0.6000000000000001 percentile value is 1944
```

```
0.7000000000000001 percentile value is 1944
0.8 percentile value is 1944
0.9 percentile value is 1945
```

In [51]:
```
sns.boxplot(mar_df['Year_Birth'])
```

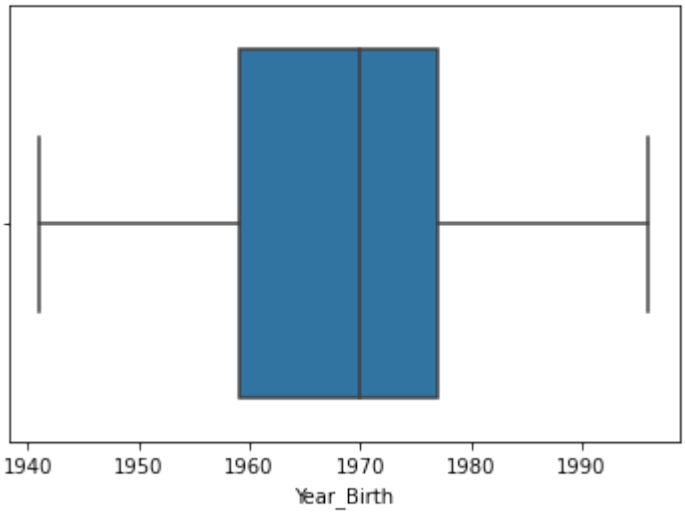
Out[51]: <AxesSubplot:xlabel='Year_Birth'>




In [52]:
```
mar_df = mar_df[mar_df['Year_Birth'] > 1940].reset_index(drop=True)
```

In [53]:
```
sns.boxplot(mar_df['Year_Birth'])
```

Out[53]: <AxesSubplot:xlabel='Year_Birth'>




In [ ]:
```

```

In [54]:
```
for i in range(0, 100, 10):
    percentile = mar_df['Income'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
0 percentile value is 1730.0
10 percentile value is 24206.0
20 percentile value is 32218.0
```

```
30 percentile value is 38361.0
40 percentile value is 44931.0
50 percentile value is 51381.5
60 percentile value is 58138.0
70 percentile value is 65106.0
80 percentile value is 71626.0
90 percentile value is 79800.0
100 percentile value is  666666.0
```

In [55]:

```
for i in range(90, 100, 1):
    percentile = mar_df['Income'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 79800.0
91 percentile value is 80398.0
92 percentile value is 81217.0
93 percentile value is 82072.0
94 percentile value is 82800.0
95 percentile value is 84117.0
96 percentile value is 85696.0
97 percentile value is 87679.0
98 percentile value is 90765.0
99 percentile value is 94472.0
100 percentile value is  666666.0
```

In [56]:

```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['Income'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

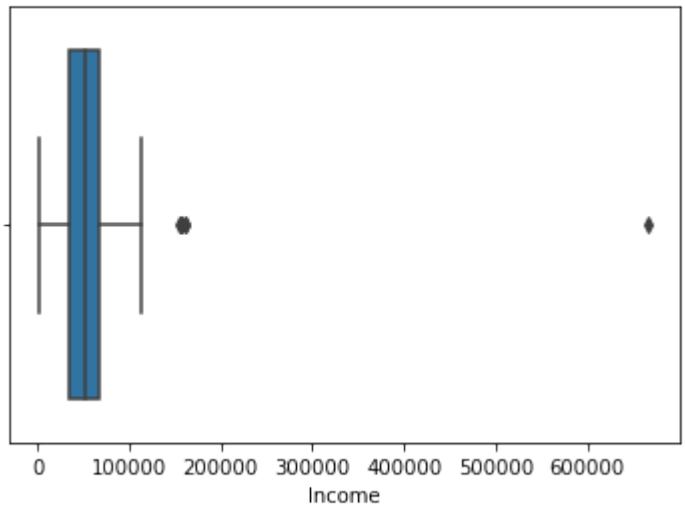
```
99.0 percentile value is 94472.0
99.1 percentile value is 94871.0
99.2 percentile value is 96547.0
99.3 percentile value is 96876.0
99.4 percentile value is 98777.0
99.5 percentile value is 102160.0
99.6 percentile value is 113734.0
99.7 percentile value is 156924.0
99.8 percentile value is 157243.0
99.9 percentile value is 160803.0
100 percentile value is  666666.0
```

In [57]:

```
sns.boxplot(mar_df['Income'])
```

Out[57]:

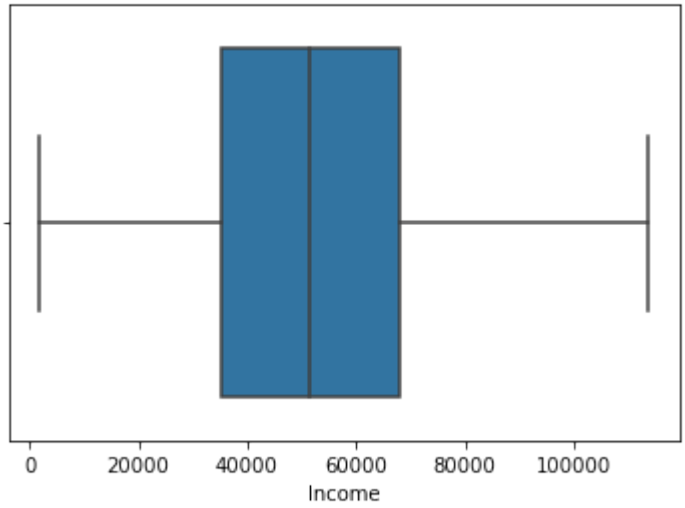
```
<AxesSubplot:xlabel='Income'>
```




In [58]:
```
mar_df = mar_df[mar_df['Income'] < 120000].reset_index(drop=True)
```

In [59]:
```
sns.boxplot(mar_df['Income'])
```

Out[59]: <AxesSubplot:xlabel='Income'>




In [ ]:

In [60]:
```
for i in range(0, 100, 10):
    percentile = mar_df['MntWines'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
0 percentile value is 0
10 percentile value is 6
20 percentile value is 16
30 percentile value is 34
40 percentile value is 84
50 percentile value is 177
60 percentile value is 291
70 percentile value is 421
80 percentile value is 583
90 percentile value is 823
100 percentile value is  1493
```

In [61]:
```
for i in range(90, 100, 1):
    percentile = mar_df['MntWines'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

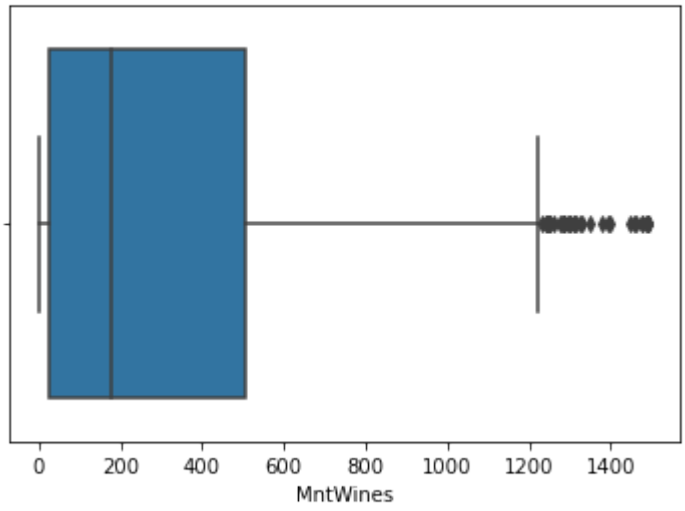
```
90 percentile value is 823
91 percentile value is 861
92 percentile value is 901
93 percentile value is 938
94 percentile value is 967
95 percentile value is 1000
96 percentile value is 1045
97 percentile value is 1111
98 percentile value is 1193
99 percentile value is 1285
100 percentile value is  1493
```

In [62]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['MntWines'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 1285
99.1 percentile value is 1296
99.2 percentile value is 1308
99.3 percentile value is 1315
99.4 percentile value is 1332
99.5 percentile value is 1379
99.6 percentile value is 1449
99.7 percentile value is 1462
99.8 percentile value is 1478
99.9 percentile value is 1492
100 percentile value is  1493
```

In [63]:
```
sns.boxplot(mar_df['MntWines'])
```

Out[63]: <AxesSubplot:xlabel='MntWines'>




- There are many datapoints which are low-level outliers. As these are real and important observations, we will keep them. We will only remove extreme outliers.

In [ ]:

In [64]:
```
for i in range(0, 100, 10):
    percentile = mar_df['MntFruits'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
0 percentile value is 0
10 percentile value is 0
20 percentile value is 1
30 percentile value is 2
40 percentile value is 4
50 percentile value is 8
60 percentile value is 15
70 percentile value is 25
80 percentile value is 44
90 percentile value is 83
100 percentile value is  199
```

In [65]:
```
for i in range(90, 100, 1):
    percentile = mar_df['MntFruits'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

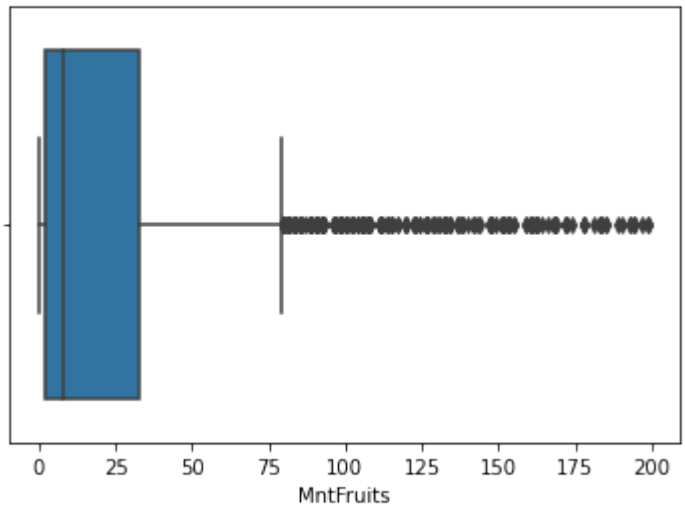
```
90 percentile value is 83
91 percentile value is 89
92 percentile value is 98
93 percentile value is 105
94 percentile value is 112
95 percentile value is 123
96 percentile value is 133
97 percentile value is 142
98 percentile value is 159
99 percentile value is 172
100 percentile value is  199
```

In [66]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['MntFruits'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 172
99.1 percentile value is 174
99.2 percentile value is 178
99.3 percentile value is 183
99.4 percentile value is 183
99.5 percentile value is 185
99.6 percentile value is 190
99.7 percentile value is 193
99.8 percentile value is 194
99.9 percentile value is 197
100 percentile value is  199
```

In [67]:
```
sns.boxplot(mar_df['MntFruits'])
```

Out[67]:
```
<AxesSubplot:xlabel='MntFruits'>
```




- There are many datapoints which are low-level outliers. As these are real and important observations, we will keep them. We will only remove extreme outliers.

In [ ]:

In [68]:
```
for i in range(90, 100, 1):
    percentile = mar_df['MntMeatProducts'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 497
91 percentile value is 520
92 percentile value is 549
93 percentile value is 573
94 percentile value is 614
95 percentile value is 685
96 percentile value is 724
97 percentile value is 774
98 percentile value is 818
99 percentile value is 899
100 percentile value is  1725
```

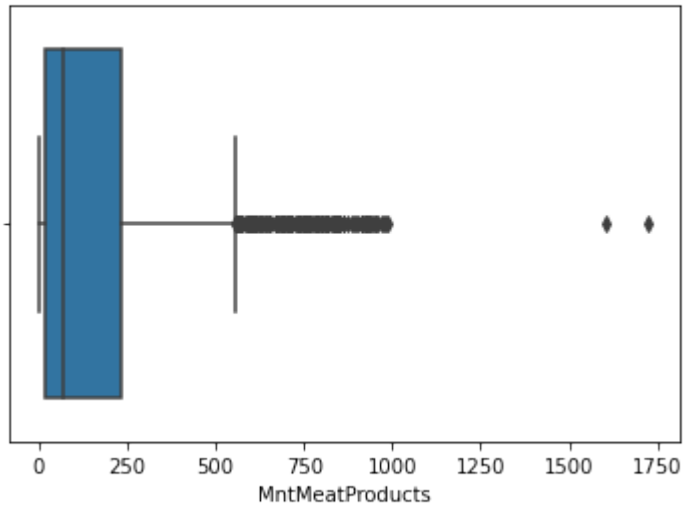
In [69]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['MntMeatProducts'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])

# >1000
```

```
99.0 percentile value is 899
99.1 percentile value is 915
99.2 percentile value is 925
99.3 percentile value is 925
99.4 percentile value is 932
99.5 percentile value is 936
99.6 percentile value is 951
99.7 percentile value is 961
99.8 percentile value is 974
99.9 percentile value is 984
100 percentile value is  1725
```

In [70]:
```
sns.boxplot(mar_df['MntMeatProducts'])
```

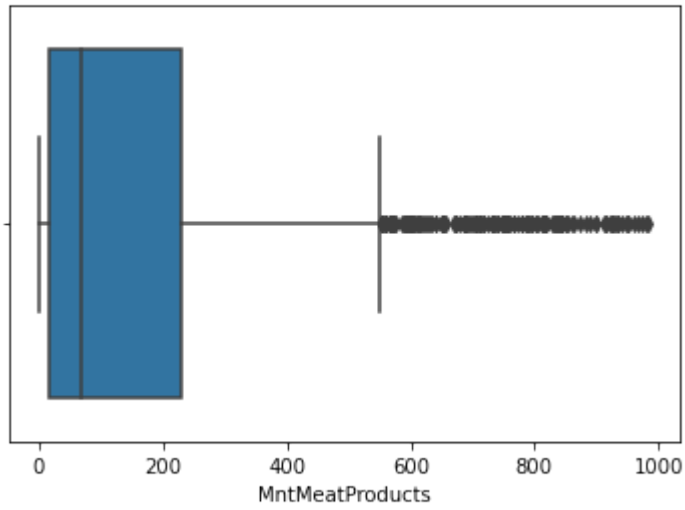
Out[70]: <AxesSubplot:xlabel='MntMeatProducts'>




In [71]:
```
mar_df = mar_df[mar_df['MntMeatProducts'] < 1000].reset_index(drop=True)
```

In [72]:
```
sns.boxplot(mar_df['MntMeatProducts'])
```

Out[72]: <AxesSubplot:xlabel='MntMeatProducts'>




- We have removed extreme outliers and kept low-level outliers which are needed for analysis

In [73]:
```
mar_df['MntMeatProducts'].quantile(q=[0.25, 0.50, 0.75, 0.9, 0.95, 1.0])
```

Out[73]:
```
0.25     16.00
0.50     67.00
0.75    230.00
0.90    493.50
0.95    673.75
1.00    984.00
Name: MntMeatProducts, dtype: float64
```

In [ ]:
```

```

In [74]:
```
for i in range(90, 100, 1):
    percentile = mar_df['MntFishProducts'].values
```

```
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

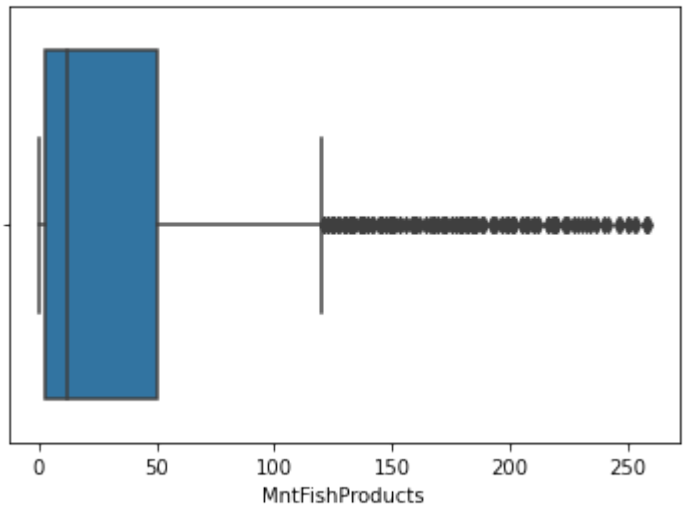
```
90 percentile value is 121
91 percentile value is 132
92 percentile value is 138
93 percentile value is 150
94 percentile value is 158
95 percentile value is 169
96 percentile value is 180
97 percentile value is 197
98 percentile value is 210
99 percentile value is 227
100 percentile value is  259
```

In [75]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['MntFishProducts'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 227
99.1 percentile value is 229
99.2 percentile value is 234
99.3 percentile value is 237
99.4 percentile value is 240
99.5 percentile value is 242
99.6 percentile value is 250
99.7 percentile value is 250
99.8 percentile value is 254
99.9 percentile value is 258
100 percentile value is  259
```

In [76]:
```
sns.boxplot(mar_df['MntFishProducts'])
```

Out[76]: <AxesSubplot:xlabel='MntFishProducts'>




In [ ]:
```
```

In [77]:
```
for i in range(90, 100, 1):
    percentile = mar_df['MntSweetProducts'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

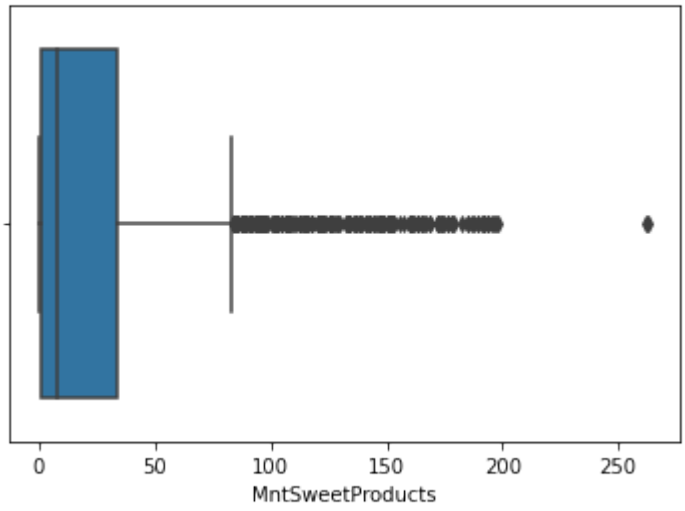
```
90 percentile value is 89
91 percentile value is 94
92 percentile value is 101
93 percentile value is 107
94 percentile value is 118
95 percentile value is 126
96 percentile value is 137
97 percentile value is 148
98 percentile value is 161
99 percentile value is 178
100 percentile value is  263
```

In [78]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['MntSweetProducts'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 178
99.1 percentile value is 179
99.2 percentile value is 185
99.3 percentile value is 188
99.4 percentile value is 189
99.5 percentile value is 192
99.6 percentile value is 194
99.7 percentile value is 194
99.8 percentile value is 196
99.9 percentile value is 198
100 percentile value is  263
```

In [79]:
```
sns.boxplot(mar_df['MntSweetProducts'])
```

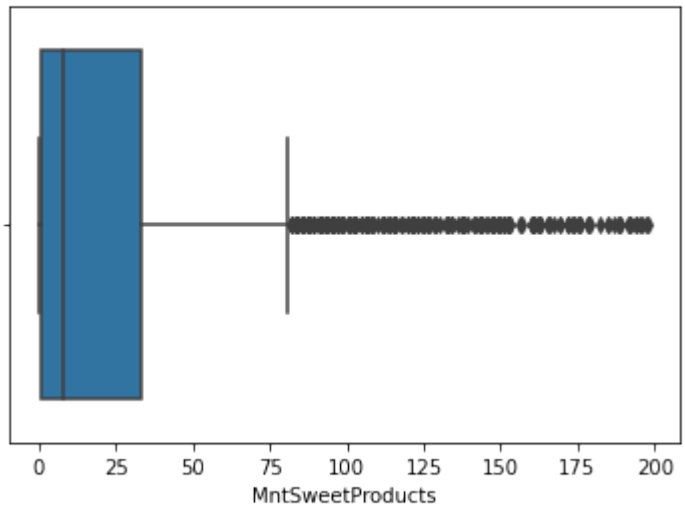
Out[79]: <AxesSubplot:xlabel='MntSweetProducts'>




In [80]:
```
mar_df = mar_df[mar_df['MntSweetProducts'] < 200].reset_index(drop=True)
```

In [81]:
```
sns.boxplot(mar_df['MntSweetProducts'])
```

Out[81]: <AxesSubplot:xlabel='MntSweetProducts'>




In [ ]:

```
```

In [82]:

```
for i in range(90, 100, 1):
    percentile = mar_df['MntGoldProds'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 122
91 percentile value is 129
92 percentile value is 135
93 percentile value is 145
94 percentile value is 152
95 percentile value is 163
96 percentile value is 174
97 percentile value is 187
98 percentile value is 200
99 percentile value is 227
100 percentile value is  321
```

In [83]:

```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['MntGoldProds'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

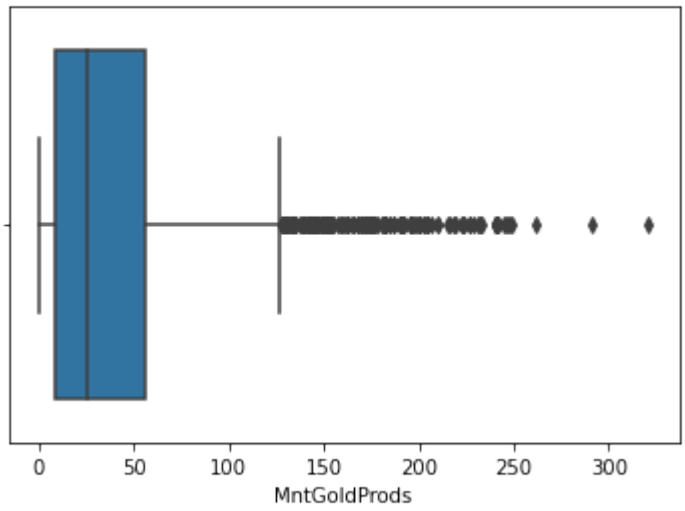
```
99.0 percentile value is 227
99.1 percentile value is 229
99.2 percentile value is 232
99.3 percentile value is 241
99.4 percentile value is 241
99.5 percentile value is 241
99.6 percentile value is 242
99.7 percentile value is 246
99.8 percentile value is 248
99.9 percentile value is 262
100 percentile value is  321
```

In [84]:

```
sns.boxplot(mar_df['MntGoldProds'])
```

Out[84]:

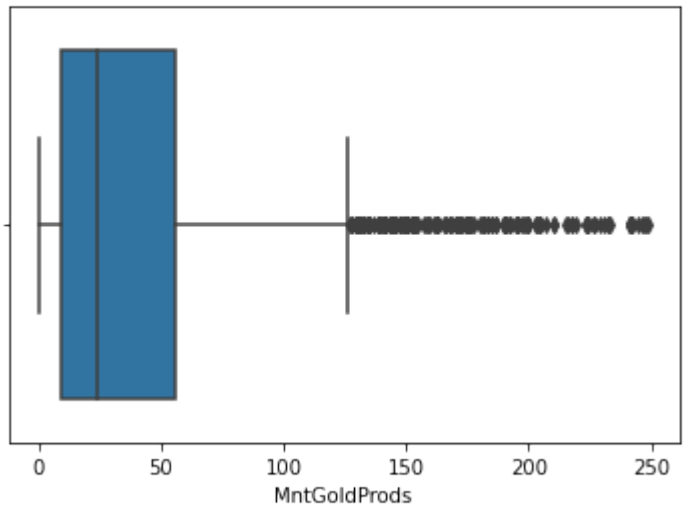
```
<AxesSubplot:xlabel='MntGoldProds'>
```




In [85]:
```
mar_df = mar_df[mar_df['MntGoldProds'] < 250].reset_index(drop=True)
```

In [86]:
```
sns.boxplot(mar_df['MntGoldProds'])
```

Out[86]: <AxesSubplot:xlabel='MntGoldProds'>




In [ ]:
```
```

In [87]:
```
for i in range(90, 100, 1):
    percentile = mar_df['NumWebPurchases'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

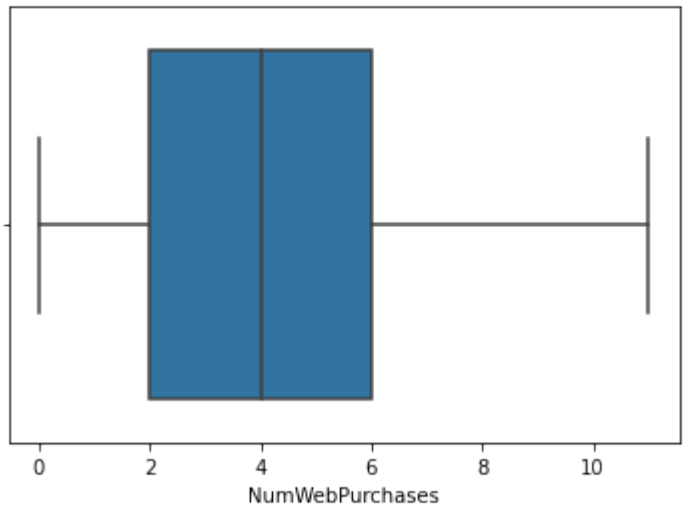
```
90 percentile value is 8
91 percentile value is 8
92 percentile value is 8
93 percentile value is 9
94 percentile value is 9
95 percentile value is 9
96 percentile value is 9
97 percentile value is 10
98 percentile value is 10
99 percentile value is 11
100 percentile value is  11
```

In [88]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['NumWebPurchases'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 11
99.1 percentile value is 11
99.2 percentile value is 11
99.3 percentile value is 11
99.4 percentile value is 11
99.5 percentile value is 11
99.6 percentile value is 11
99.7 percentile value is 11
99.8 percentile value is 11
99.9 percentile value is 11
100 percentile value is  11
```

In [89]:
```
sns.boxplot(mar_df['NumWebPurchases'])
```

Out[89]: <AxesSubplot:xlabel='NumWebPurchases'>




In [ ]:
```
```

In [90]:
```
for i in range(90, 100, 1):
    percentile = mar_df['NumCatalogPurchases'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 7
91 percentile value is 7
92 percentile value is 7
93 percentile value is 8
94 percentile value is 8
95 percentile value is 8
96 percentile value is 9
97 percentile value is 10
98 percentile value is 10
99 percentile value is 10
100 percentile value is  11
```

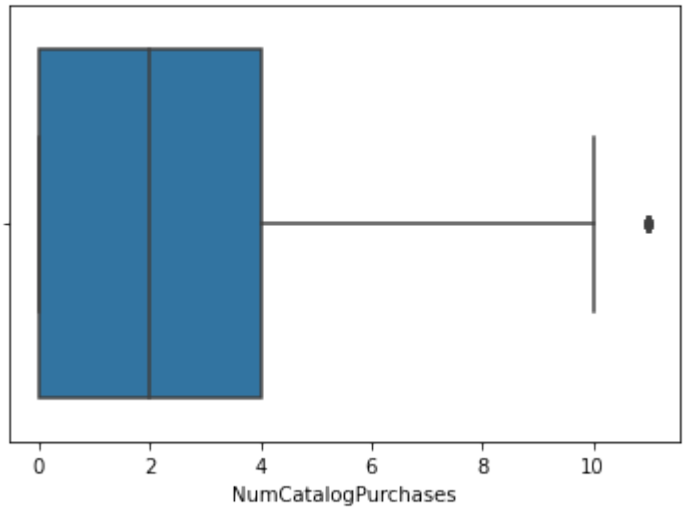
In [91]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['NumCatalogPurchases'].values
```

```
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 10
99.1 percentile value is 10
99.2 percentile value is 11
99.3 percentile value is 11
99.4 percentile value is 11
99.5 percentile value is 11
99.6 percentile value is 11
99.7 percentile value is 11
99.8 percentile value is 11
99.9 percentile value is 11
100 percentile value is  11
```

In [92]:
```
sns.boxplot(mar_df['NumCatalogPurchases'])
```

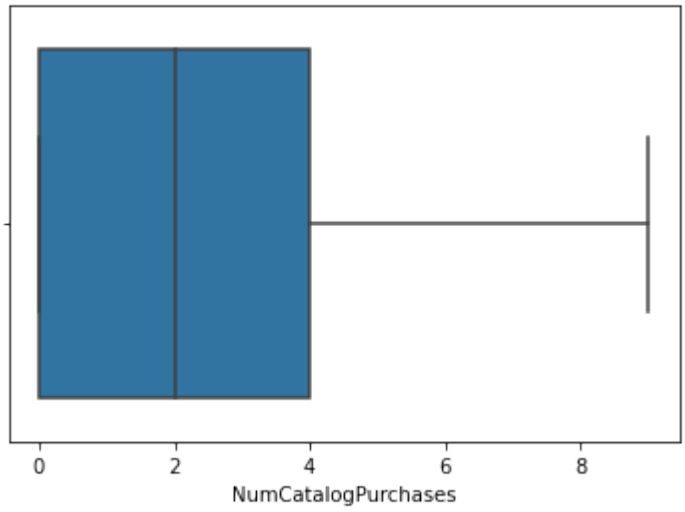
Out[92]: <AxesSubplot:xlabel='NumCatalogPurchases'>




In [93]:
```
mar_df = mar_df[mar_df['NumCatalogPurchases'] < 10].reset_index(drop=True)
```

In [94]:
```
sns.boxplot(mar_df['NumCatalogPurchases'])
```

Out[94]: <AxesSubplot:xlabel='NumCatalogPurchases'>




In [ ]:

In [95]:
```
for i in range(90, 100, 1):
    percentile = mar_df['NumStorePurchases'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

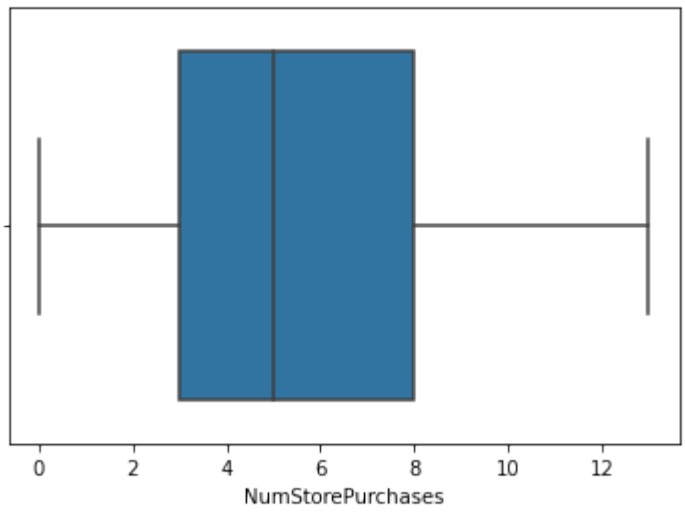
```
90 percentile value is 11
91 percentile value is 11
92 percentile value is 12
93 percentile value is 12
94 percentile value is 12
95 percentile value is 12
96 percentile value is 12
97 percentile value is 13
98 percentile value is 13
99 percentile value is 13
100 percentile value is  13
```

In [96]:
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['NumStorePurchases'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 13
99.1 percentile value is 13
99.2 percentile value is 13
99.3 percentile value is 13
99.4 percentile value is 13
99.5 percentile value is 13
99.6 percentile value is 13
99.7 percentile value is 13
99.8 percentile value is 13
99.9 percentile value is 13
100 percentile value is  13
```

In [97]:
```
sns.boxplot(mar_df['NumStorePurchases'])
```

Out[97]: <AxesSubplot:xlabel='NumStorePurchases'>




In [ ]:

In [98]:
```
for i in range(90, 100, 1):
    percentile = mar_df['NumWebVisitsMonth'].values
```

```
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 8
91 percentile value is 8
92 percentile value is 8
93 percentile value is 8
94 percentile value is 8
95 percentile value is 8
96 percentile value is 9
97 percentile value is 9
98 percentile value is 9
99 percentile value is 9
100 percentile value is  20
```

In [99]:

```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['NumWebVisitsMonth'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

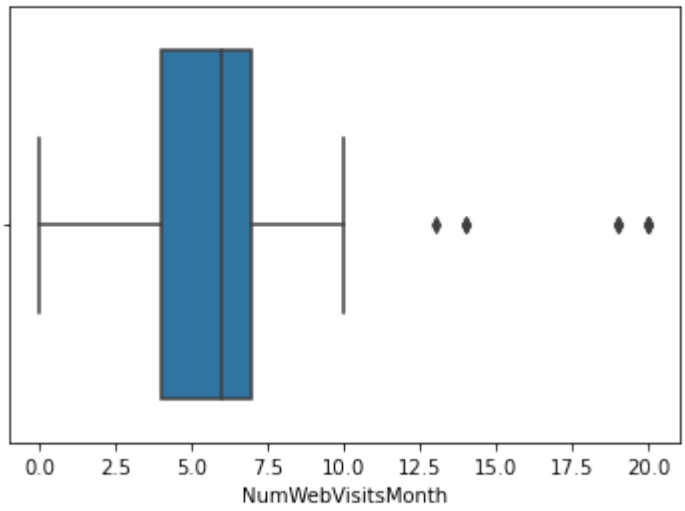
```
99.0 percentile value is 9
99.1 percentile value is 9
99.2 percentile value is 9
99.3 percentile value is 9
99.4 percentile value is 9
99.5 percentile value is 10
99.6 percentile value is 10
99.7 percentile value is 14
99.8 percentile value is 19
99.9 percentile value is 20
100 percentile value is  20
```

In [100…

```
sns.boxplot(mar_df['NumWebVisitsMonth'])
```

Out[100…

```
<AxesSubplot:xlabel='NumWebVisitsMonth'>
```




In [101…

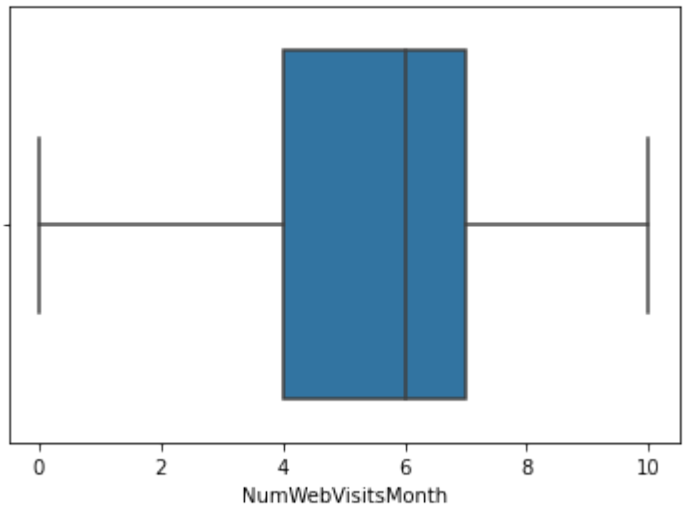
```
mar_df = mar_df[mar_df['NumWebVisitsMonth'] < 13].reset_index(drop=True)
```

In [102…

```
sns.boxplot(mar_df['NumWebVisitsMonth'])
```

Out[102…

```
<AxesSubplot:xlabel='NumWebVisitsMonth'>
```




In [103…
```
mar_df.columns
```

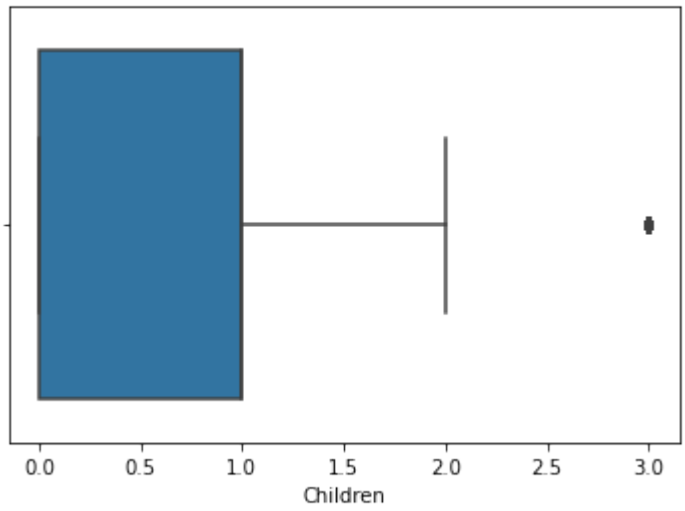
Out[103…
```
Index(['Year_Birth', 'Education', 'Marital_Status', 'Income', 'Recency',
       'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
       'AcceptedCmp1', 'AcceptedCmp2', 'Response', 'Complain', 'Country',
       'Children', 'Year_Customer', 'TotalMntSpent', 'TotalPurchases',
       'TotalCmpAccepted'],
      dtype='object')
```

In [104…
```
for i in range(90, 100, 1):
    percentile = mar_df['Children'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 2
91 percentile value is 2
92 percentile value is 2
93 percentile value is 2
94 percentile value is 2
95 percentile value is 2
96 percentile value is 2
97 percentile value is 2
98 percentile value is 3
99 percentile value is 3
100 percentile value is  3
```

In [105…
```
sns.boxplot(mar_df['Children'])
```

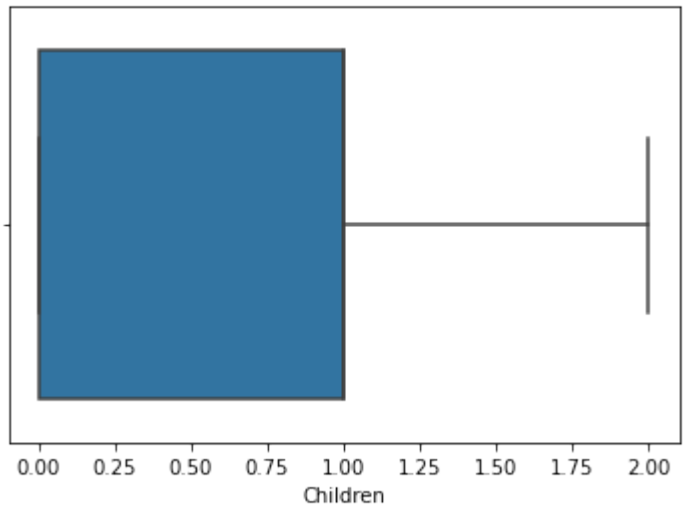
Out[105…
```
<AxesSubplot:xlabel='Children'>
```




In [106…
```
mar_df = mar_df[mar_df['Children'] < 3].reset_index(drop=True)
```

In [107…
```
sns.boxplot(mar_df['Children'])
```

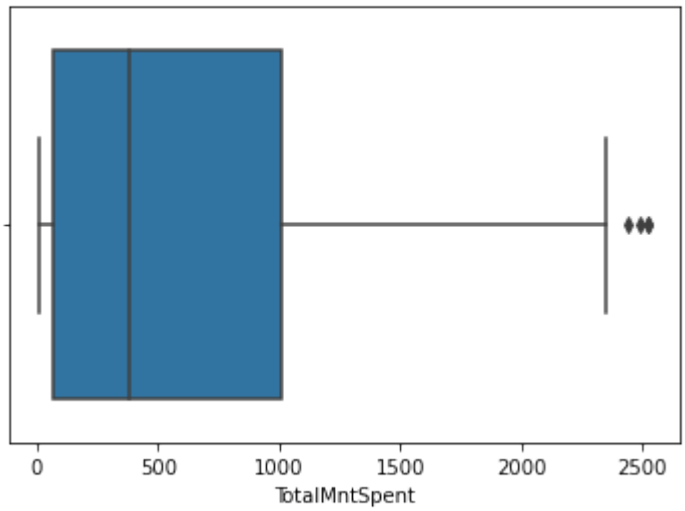
Out[107… <AxesSubplot:xlabel='Children'>




In [ ]:

In [108…
```
sns.boxplot(mar_df['TotalMntSpent'])
```

Out[108… <AxesSubplot:xlabel='TotalMntSpent'>




In [109…

```
for i in range(90, 100, 1):
    percentile = mar_df['TotalMntSpent'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 1501
91 percentile value is 1555
92 percentile value is 1597
93 percentile value is 1650
94 percentile value is 1690
95 percentile value is 1753
96 percentile value is 1820
97 percentile value is 1918
98 percentile value is 2008
99 percentile value is 2116
100 percentile value is  2525
```

In [110…

```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['TotalMntSpent'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 2116
99.1 percentile value is 2153
99.2 percentile value is 2194
99.3 percentile value is 2211
99.4 percentile value is 2231
99.5 percentile value is 2279
99.6 percentile value is 2302
99.7 percentile value is 2346
99.8 percentile value is 2352
99.9 percentile value is 2486
100 percentile value is  2525
```

In [111…

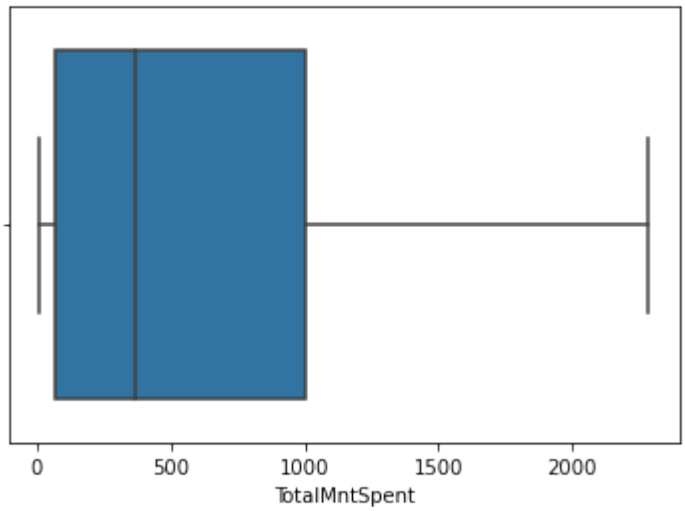
```
mar_df = mar_df[mar_df['TotalMntSpent'] < 2300].reset_index(drop=True)
```

In [112…

```
sns.boxplot(mar_df['TotalMntSpent'])
```

Out[112…

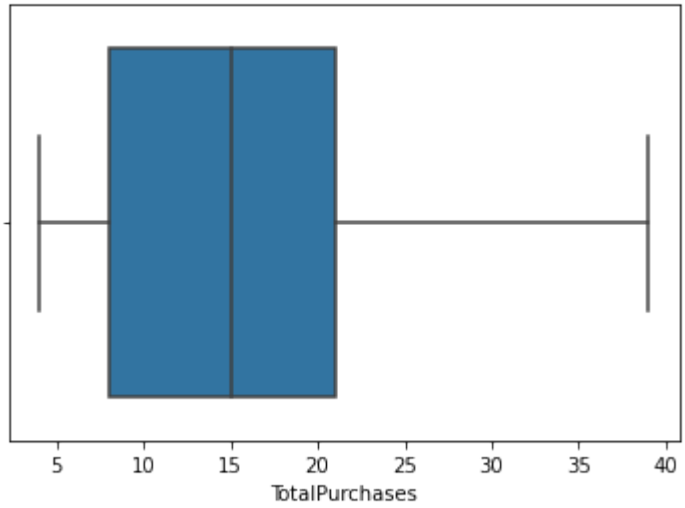
```
<AxesSubplot:xlabel='TotalMntSpent'>
```




In [ ]:

In [113…
```
sns.boxplot(mar_df['TotalPurchases'])
```

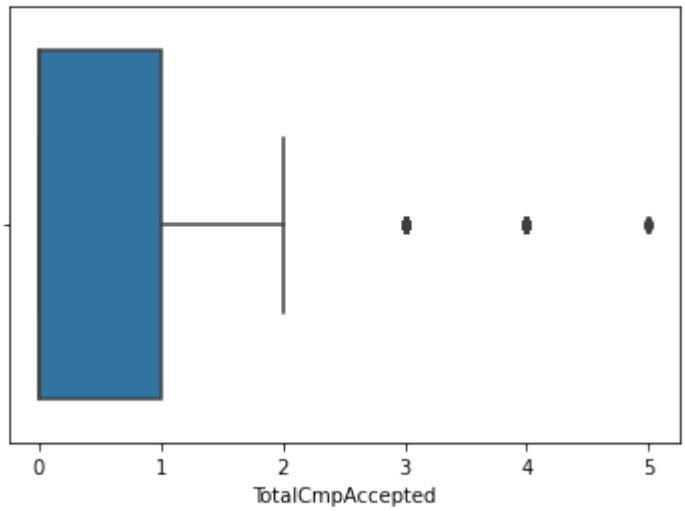
Out[113… <AxesSubplot:xlabel='TotalPurchases'>




In [ ]:

In [114…
```
sns.boxplot(mar_df['TotalCmpAccepted'])
```

Out[114… <AxesSubplot:xlabel='TotalCmpAccepted'>




In [115…
```
for i in range(90, 100, 1):
    percentile = mar_df['TotalCmpAccepted'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(i,percentile[int(len(percentile)*(float
print ("100 percentile value is ",percentile[-1])
```

```
90 percentile value is 1
91 percentile value is 2
92 percentile value is 2
93 percentile value is 2
94 percentile value is 2
95 percentile value is 2
96 percentile value is 2
97 percentile value is 3
98 percentile value is 3
99 percentile value is 4
100 percentile value is  5
```

In [116…
```
for i in np.arange(0.0, 1.0, 0.1):
    percentile = mar_df['TotalCmpAccepted'].values
    percentile = np.sort(percentile, axis=None)
    print("{} percentile value is {}".format(99+i,percentile[int(len(percentile)*(fl
print ("100 percentile value is ",percentile[-1])
```

```
99.0 percentile value is 4
99.1 percentile value is 4
99.2 percentile value is 4
99.3 percentile value is 4
99.4 percentile value is 4
99.5 percentile value is 4
99.6 percentile value is 4
99.7 percentile value is 5
99.8 percentile value is 5
99.9 percentile value is 5
100 percentile value is  5
```

In [117…
```
mar_df[mar_df['TotalCmpAccepted'] == 3].count().unique()
```

Out[117…
```
array([44], dtype=int64)
```

In [118…
```
mar_df[mar_df['TotalCmpAccepted'] == 4].count().unique()
```

Out[118…
```
array([25], dtype=int64)
```

In [119…
```
mar_df[mar_df['TotalCmpAccepted'] == 5].count().unique()
```

Out[119…
```
array([8], dtype=int64)
```

- We will keep all of them all since there is not much difference between values' 3 and 5

In [ ]:
```

```

# Uni-Variate Analysis

- Let us understand the distributions of each feature

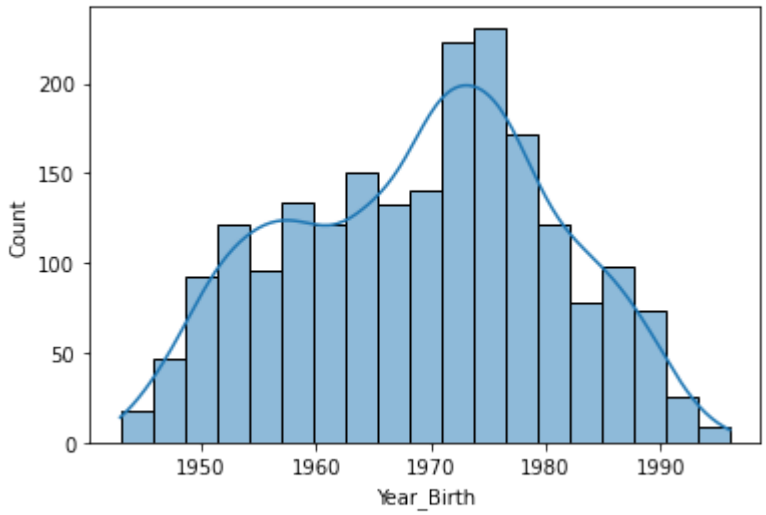
In [159…
```
mar_df.columns
```

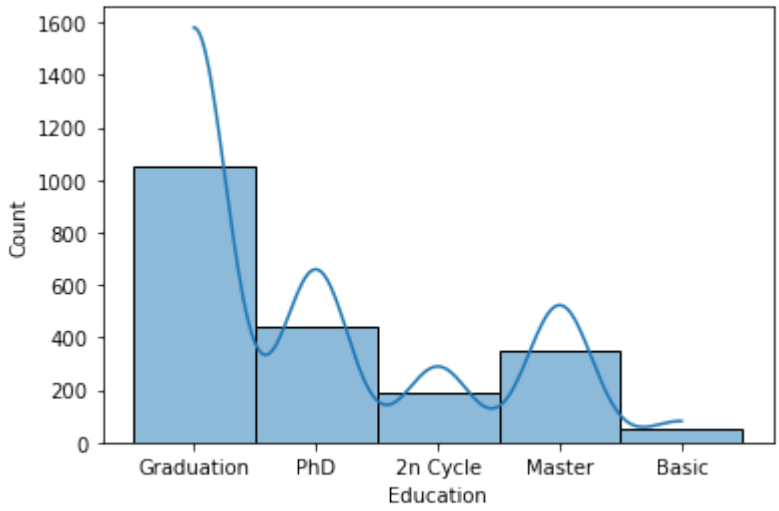
Out[159…
```
Index(['Year_Birth', 'Education', 'Marital_Status', 'Income', 'Recency',
       'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
       'AcceptedCmp1', 'AcceptedCmp2', 'Response', 'Complain', 'Country',
       'Children', 'Year_Customer', 'TotalMntSpent', 'TotalPurchases',
       'TotalCmpAccepted'],
      dtype='object')
```

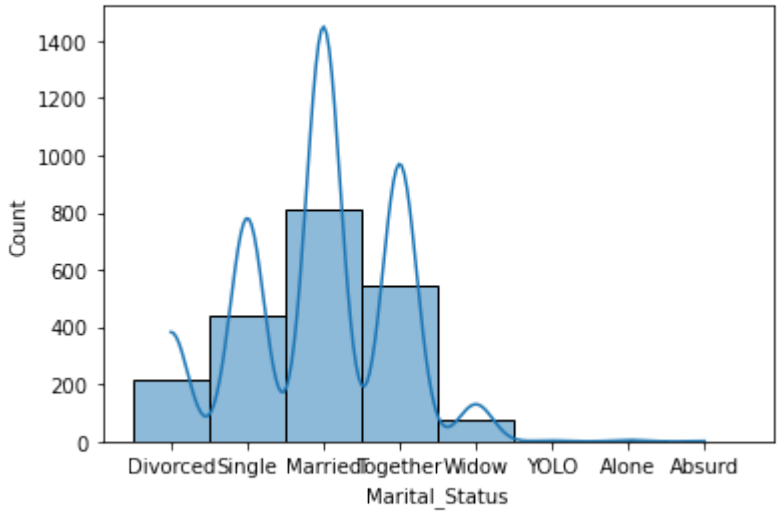
## Histogram and Probability Density Function (PDF)

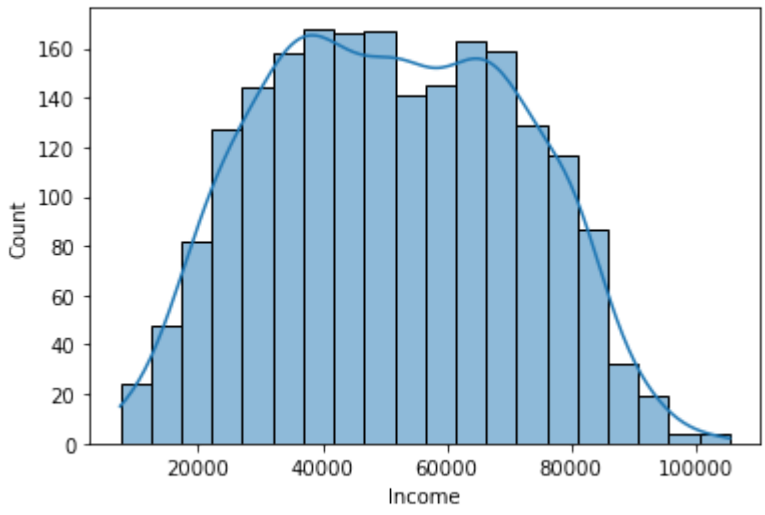
In [160…
```
df_to_plot = mar_df.drop(columns=['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'A
```

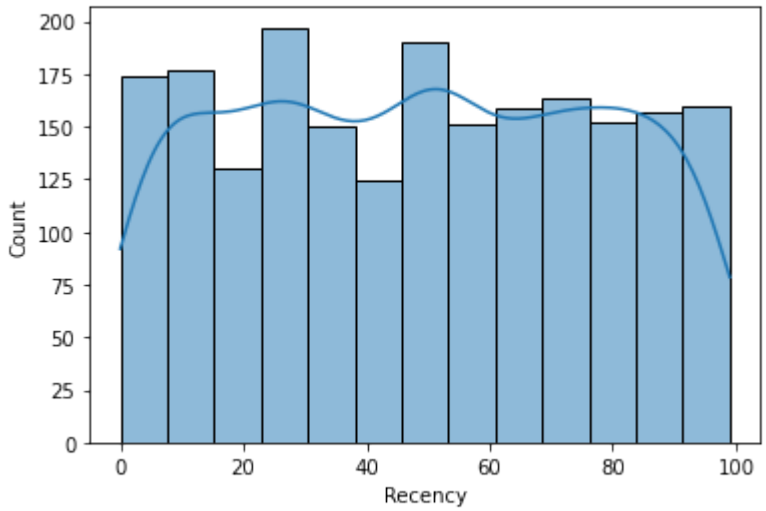
In [161…
```
for column in df_to_plot:
    sns.histplot(data=mar_df[column], kde=True)
    plt.xlabel(column)
    plt.show()
```

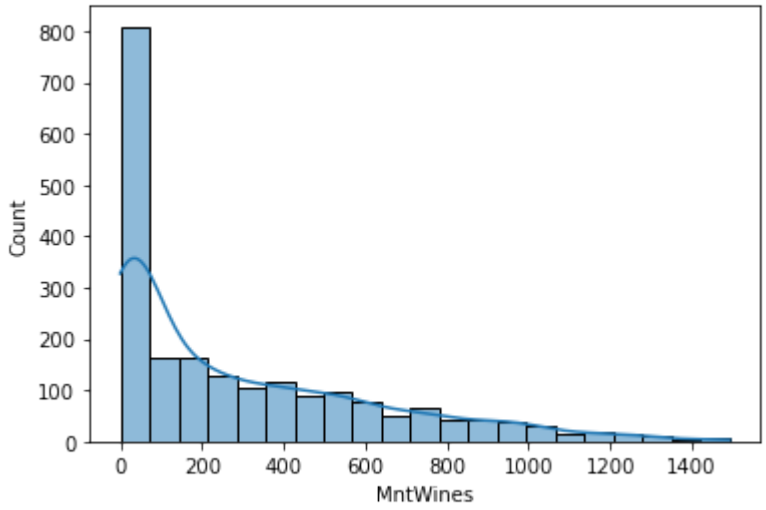


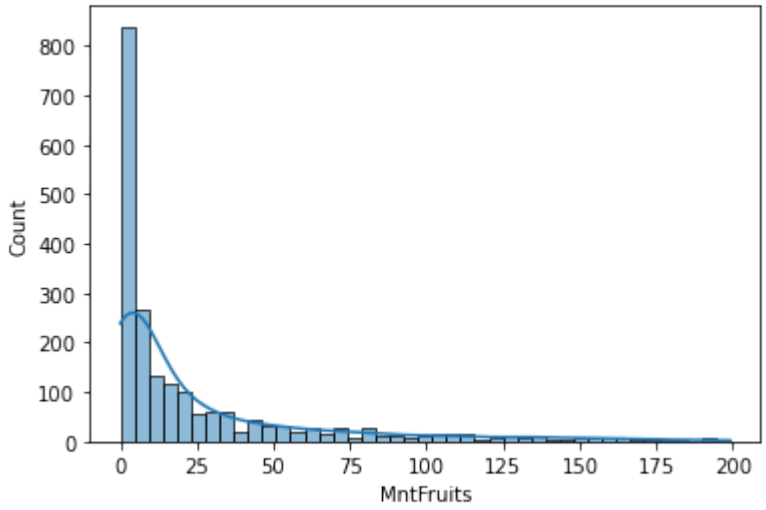

1600
1400
1200
1000
800
600
400
200
0
Count
Graduation
PhD
2n Cycle
Master
Basic
Education

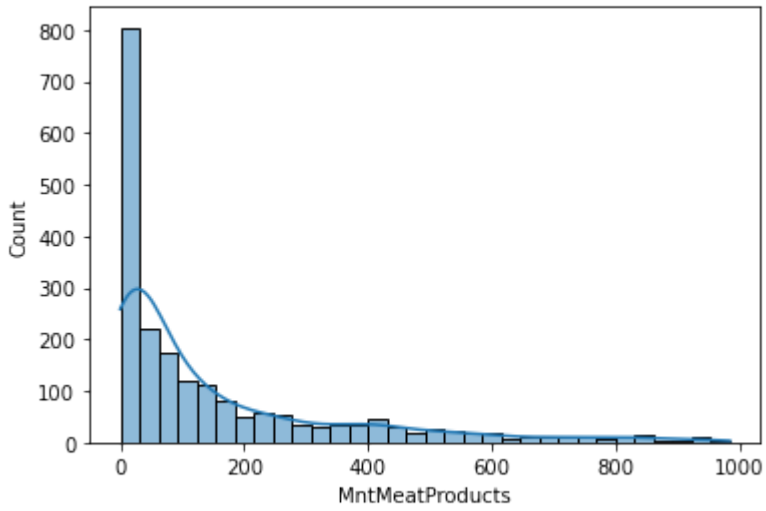

1400
1200
1000
800
600
400
200
0
Count
Divorced
Single
Married
Together
Widow
YOLO
Alone
Absurd
Marital_Status

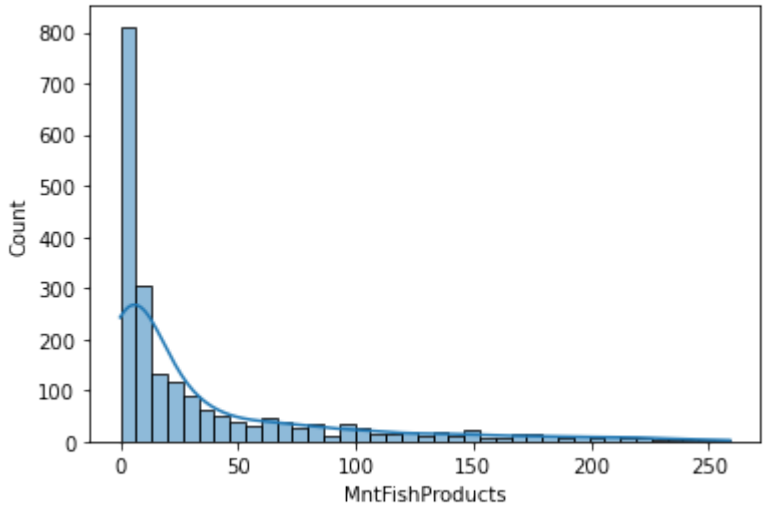

160
140
120
100
80
60
40
20
0
Count
20000
40000
60000
80000
100000
Income

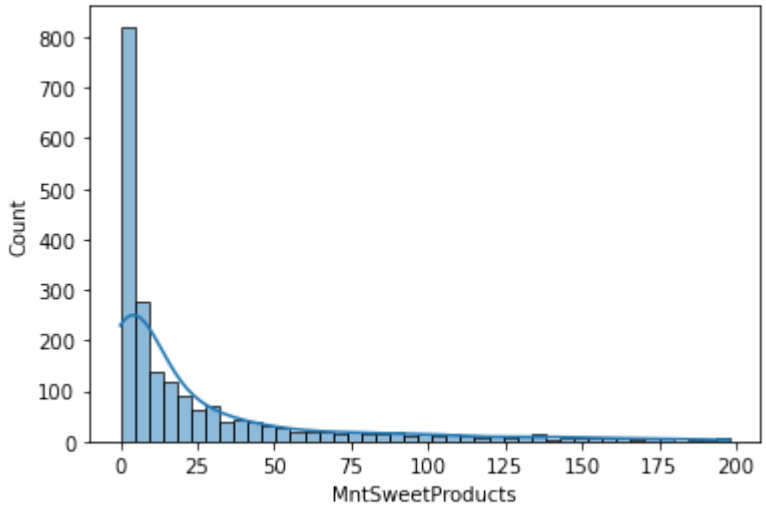

Count
Recency
0
20
40
60
80
100
0
25
50
75
100
125
150
175
200

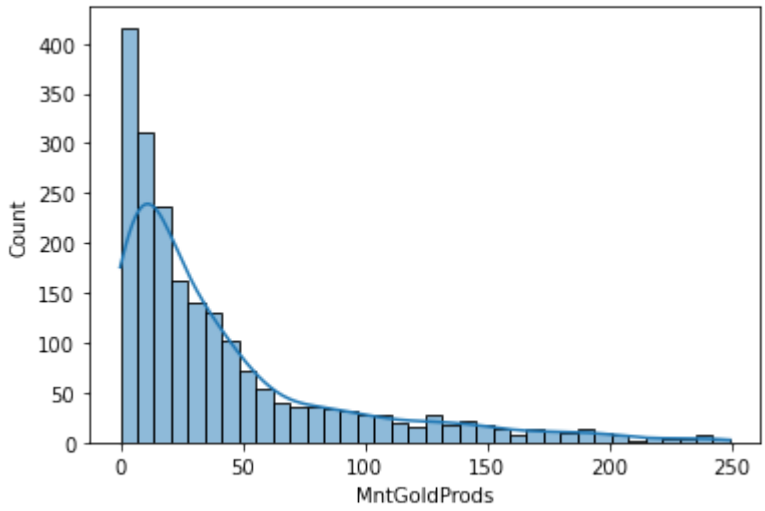

Count
MntWines
0
200
400
600
800
1000
1200
1400
0
100
200
300
400
500
600
700
800

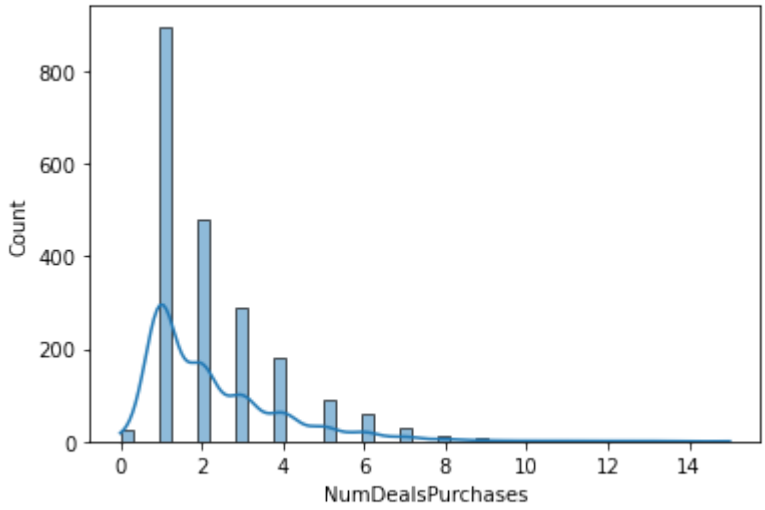

Count
MntFruits
0
25
50
75
100
125
150
175
200
0
100
200
300
400
500
600
700
800

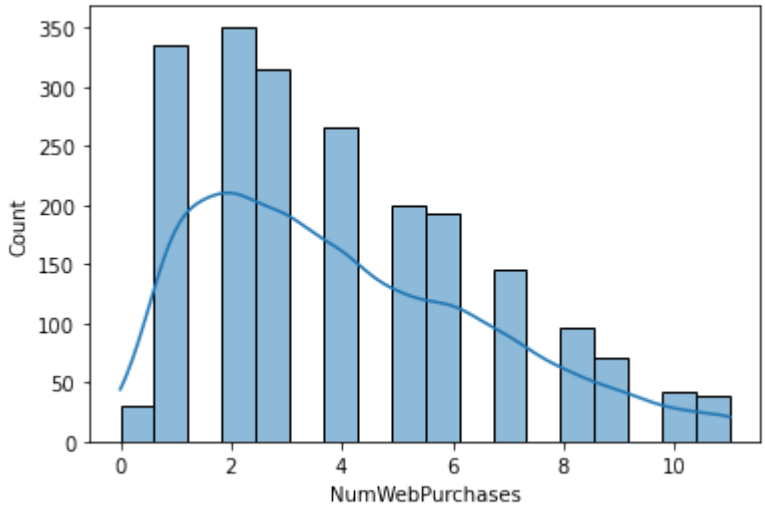

Count
MntMeatProducts

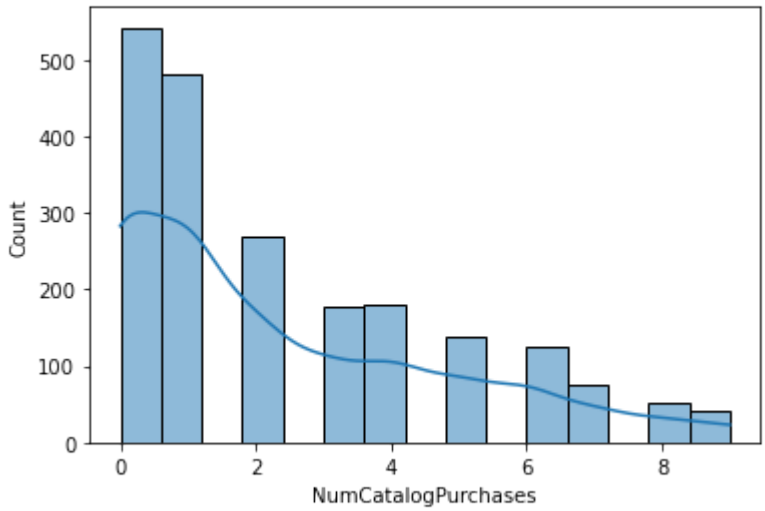

Count
MntFishProducts

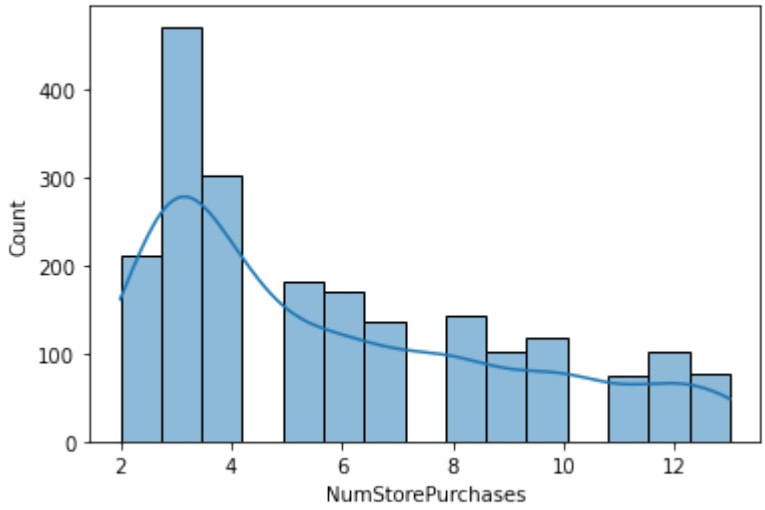

Count
MntSweetProducts

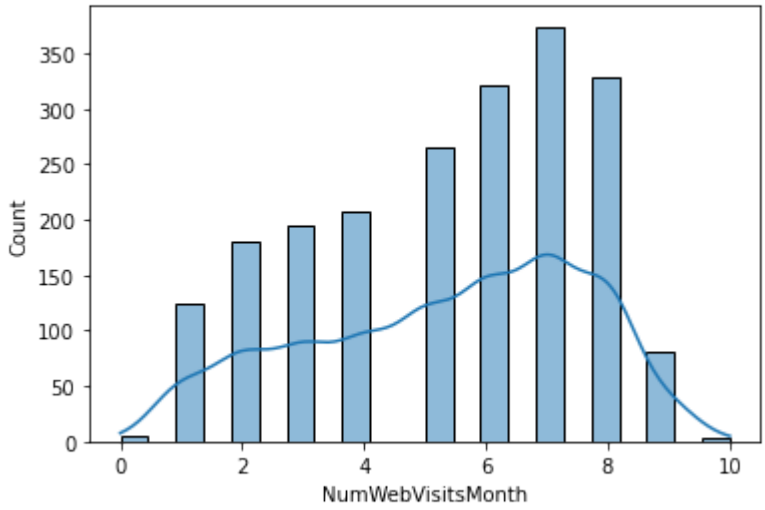

Count
MntGoldProds

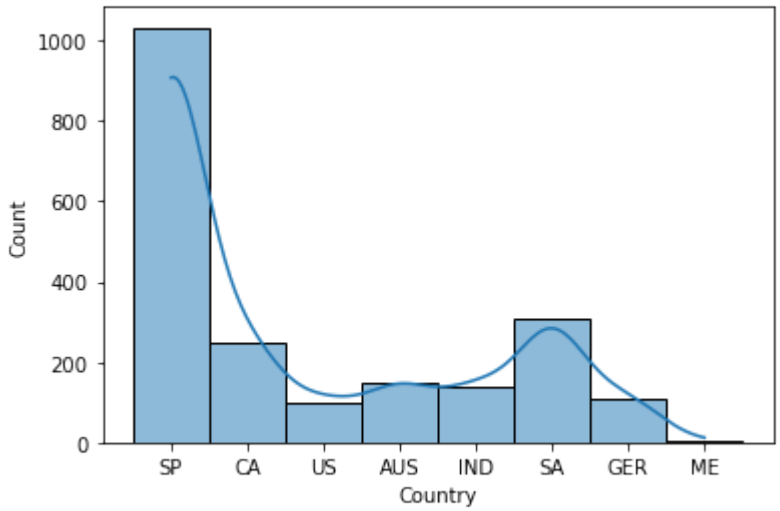

Count
NumDealsPurchases

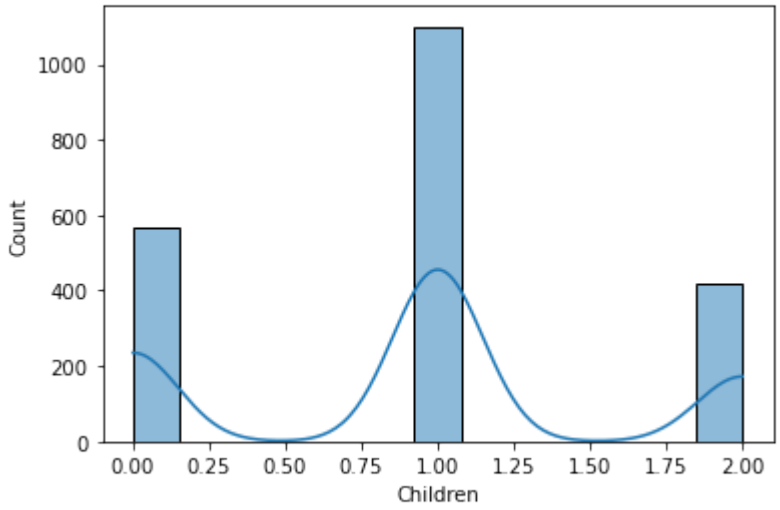

Count
NumWebPurchases

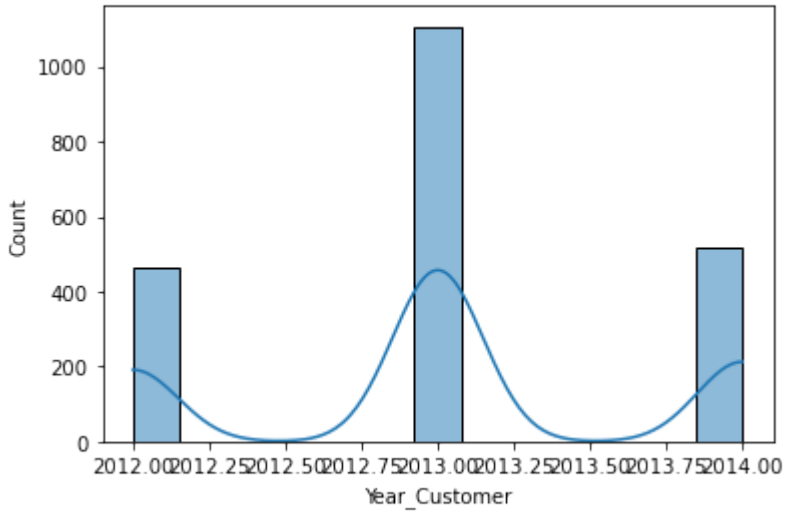

Count
NumCatalogPurchases

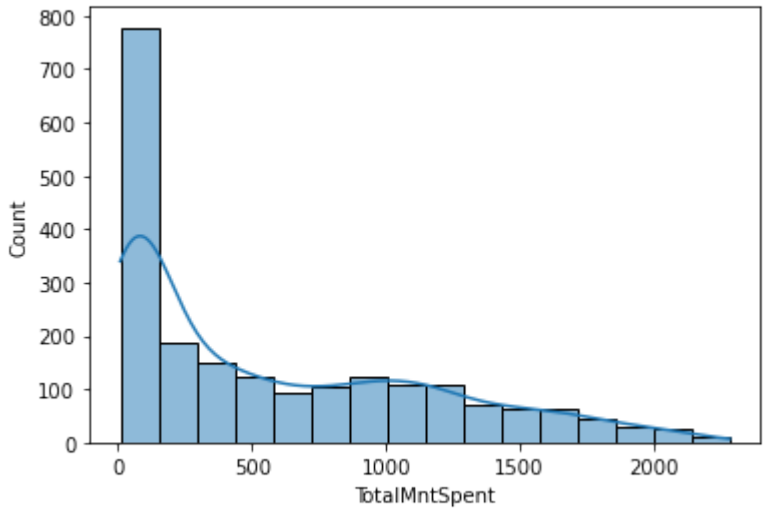

Count
NumStorePurchases

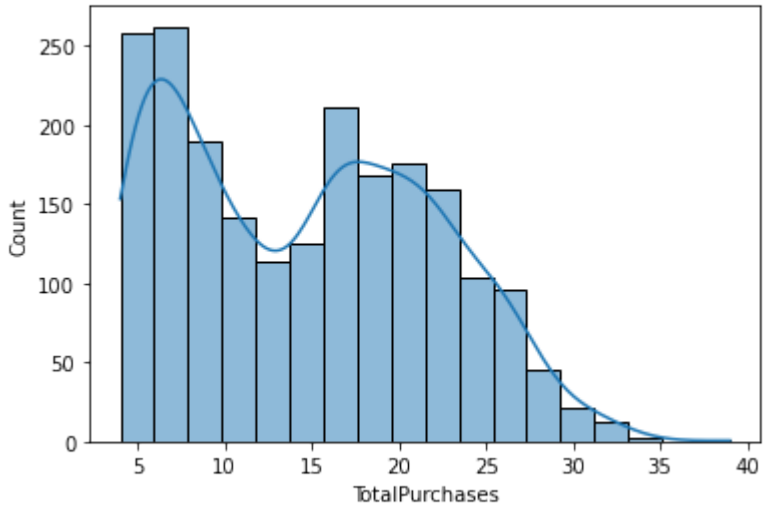

Count
NumWebVisitsMonth

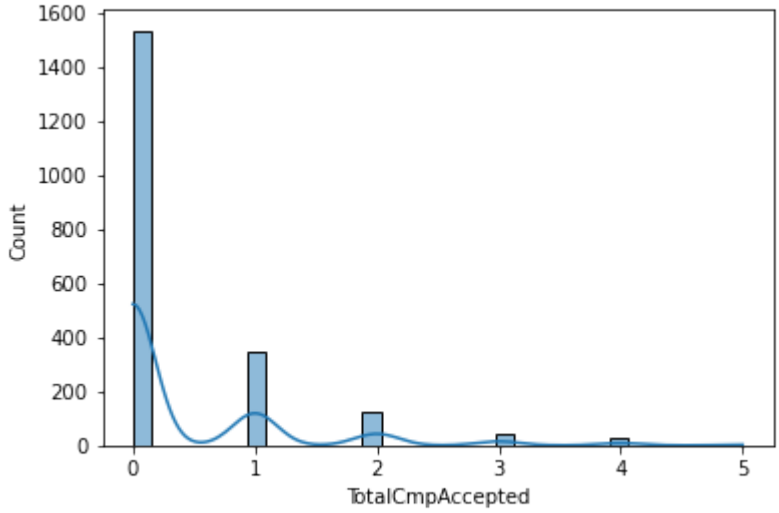

1000
800
600
400
200
0
Count
SP
CA
US
AUS
IND
SA
GER
ME
Country


1000
800
600
400
200
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Children


1000
800
600
400
200
0
Count
2012.00
2012.25
2012.50
2012.75
2013.00
2013.25
2013.50
2013.75
2014.00
Year_Customer




**Important Observations:**

- Education: Majority of customers' have a Graduate degree - 1052
- Income: Income is distributed between $(0-100000)
- MntWines: Right-Skewed distribution. Around 800 customers spent between $(0-70)
- MntFruits: Right-Skewed distribution. Around 1100 customers spent between $(0-10)
- MntMeatProducts: Right-Skewed distribution. Around 1200 customers spent between $(0-100)
- MntFishProducts: Right-Skewed distribution. Around 1100 customers spent between $(0-15)
- MntSweetProducts: Right-Skewed distribution. Around 1100 customers spent between $(0-10)

- MntGoldProds: Right-Skewed distribution. Around 1000 customers spent between $(0-25)

  MntWines, MntFruits, MntMeatProduct, MntFishProducts, MntSweetProducts, MntGoldProds follow a Log-normal distribution. That implies, there are large number of customers who have spent less-medium amount, and few number of customers who have spent high amount on these products. As the standard deviation is increasing, the tail of the distribution is also increasing.

- 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases' are all Right-Skewed distributions'.
- 'NumWebVisitsMonth' is Left-Skewed distribution
- Country: Majority of customers' are from SP (Spain) with 1032 customers'
- Children: About 1100 customers' have 1 children
- TotalMntSpent: Right-Skewed distribution. Around 1000 customers' spent between $(0-320)
- TotalCmpAccepted: Right-Skewed distribution. Around 1500 customers' did not accept any campaign

In [171…
```
mar_df[mar_df['Education'] == 'Graduation'].count().unique()
```

Out[171…
```
array([1052], dtype=int64)
```

In [201…
```
mar_df[mar_df['MntWines'] < 70].count().unique()
```

Out[201…
```
array([1107], dtype=int64)
```

In [202…
```
mar_df[mar_df['MntFruits'] < 10].count().unique()
```

Out[202…
```
array([1107], dtype=int64)
```

In [182…
```
mar_df[mar_df['MntMeatProducts'] < 100].count().unique()
```

Out[182…
```
array([1221], dtype=int64)
```

In [184…
```
mar_df[mar_df['MntFishProducts'] < 15].count().unique()
```

Out[184…
```
array([1114], dtype=int64)
```

In [186…
```
mar_df[mar_df['MntSweetProducts'] < 10].count().unique()
```

Out[186…
```
array([1098], dtype=int64)
```

In [200…
```
mar_df[mar_df['MntGoldProds'] < 25].count().unique()
```

Out[200…
```
array([1055], dtype=int64)
```

In [203…
```
mar_df[mar_df['Country'] == 'SP'].count().unique()
```

Out[203… array([1032], dtype=int64)

In [204…
```
mar_df[mar_df['Children'] == 1].count().unique()
```

Out[204… array([1100], dtype=int64)

In [213…
```
mar_df[mar_df['TotalMntSpent'] < 320].count().unique()
```

Out[213… array([1003], dtype=int64)

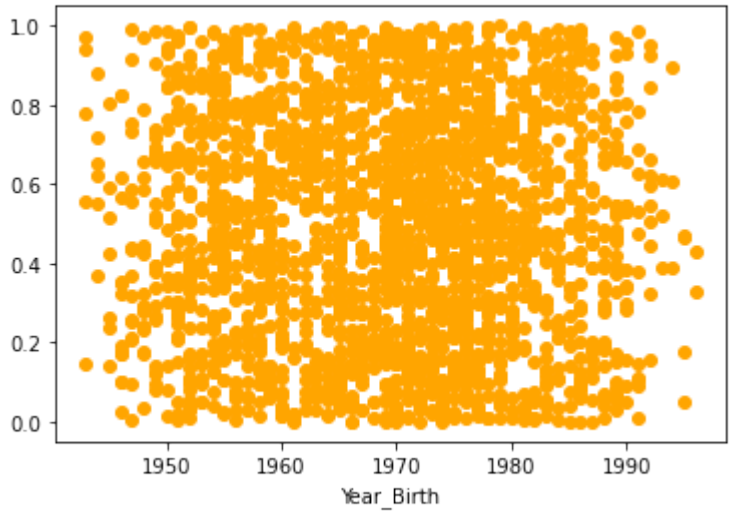
In [214…
```
mar_df[mar_df['TotalCmpAccepted'] == 0].count().unique()
```

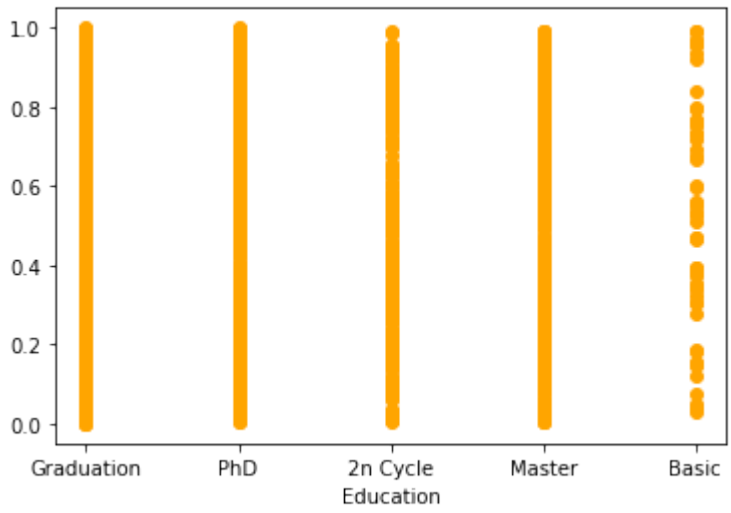
Out[214… array([1532], dtype=int64)

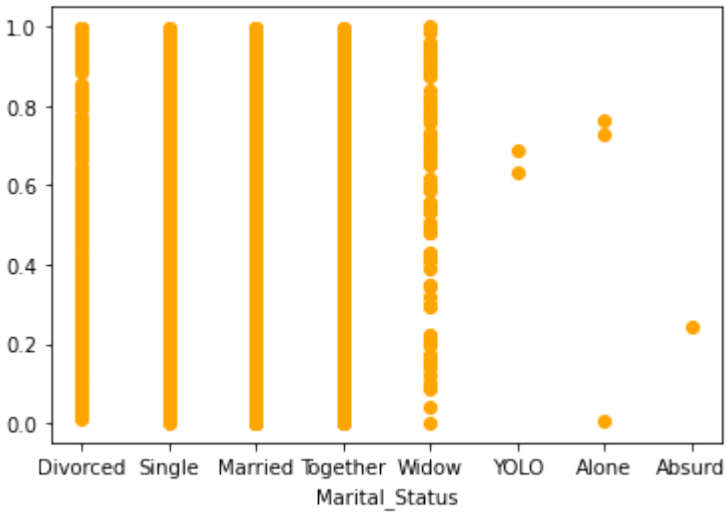
In [ ]:

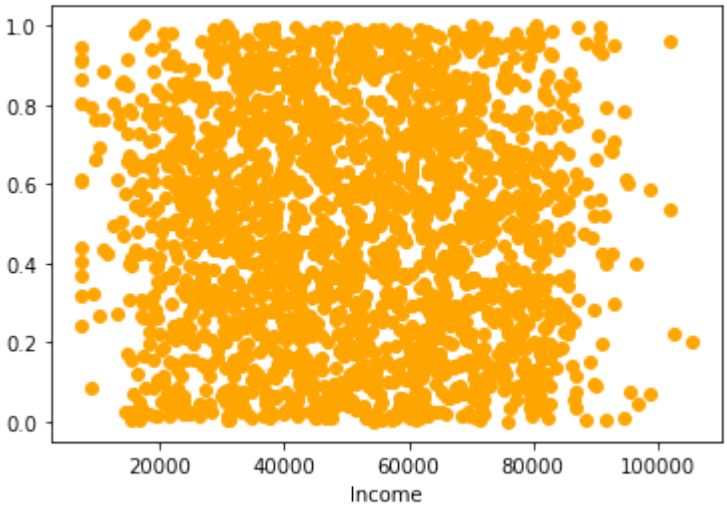
## 1-D Scatter Plot

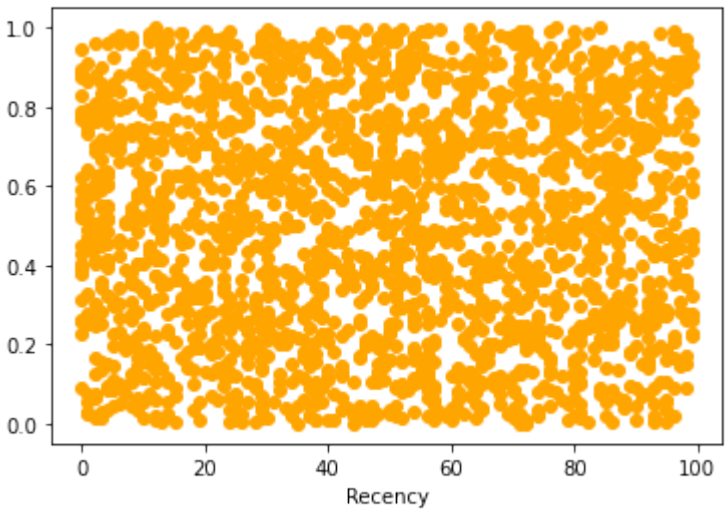
In [215…
```
for column in df_to_plot:
    plt.scatter(x = mar_df[column], y = np.random.rand(len(mar_df)), c='orange')
    plt.xlabel(column)
    plt.show()
```

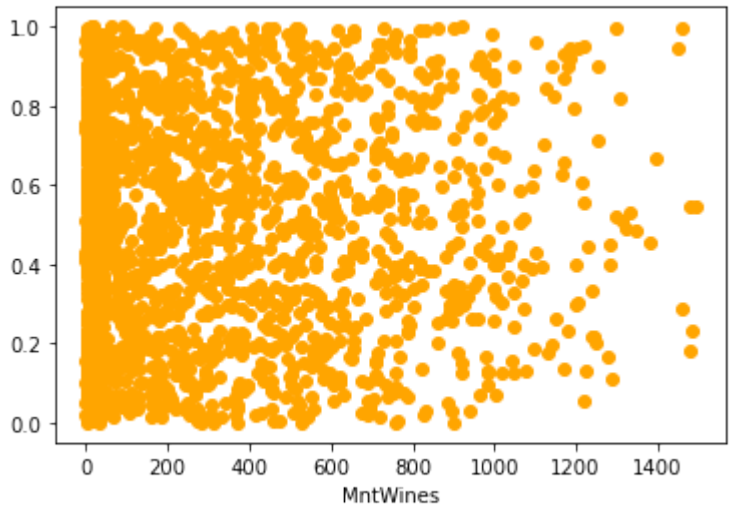


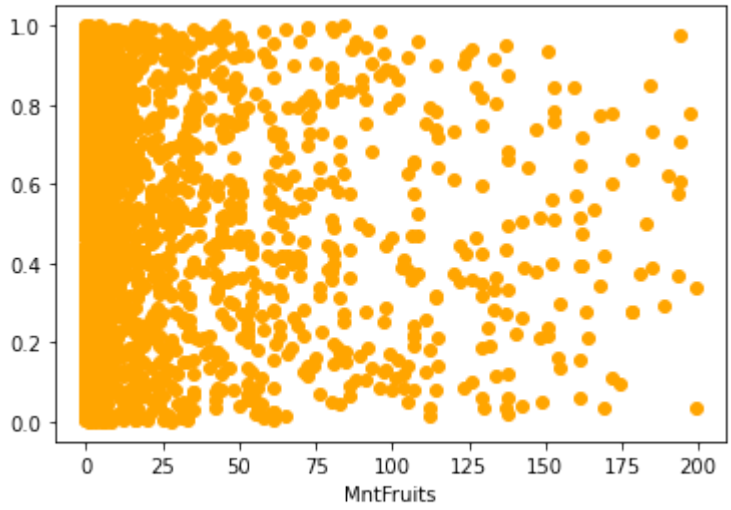



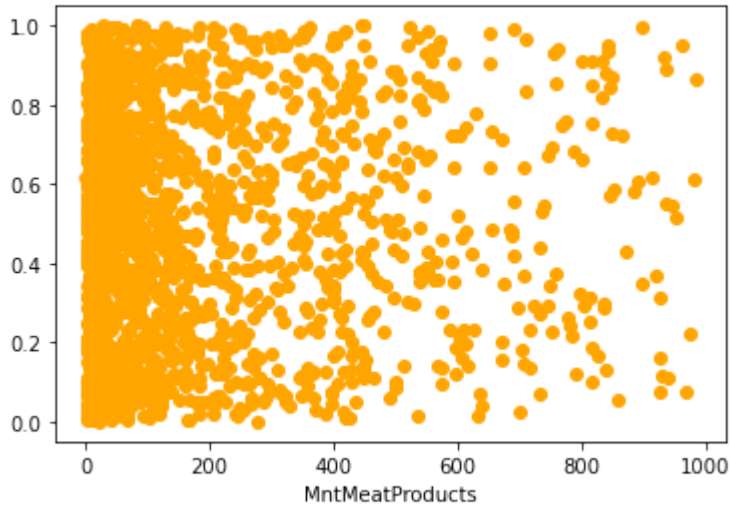

1.0
0.8
0.6
0.4
0.2
0.0
Divorced
Single
Married
Together
Widow
YOLO
Alone
Absurd
Marital_Status

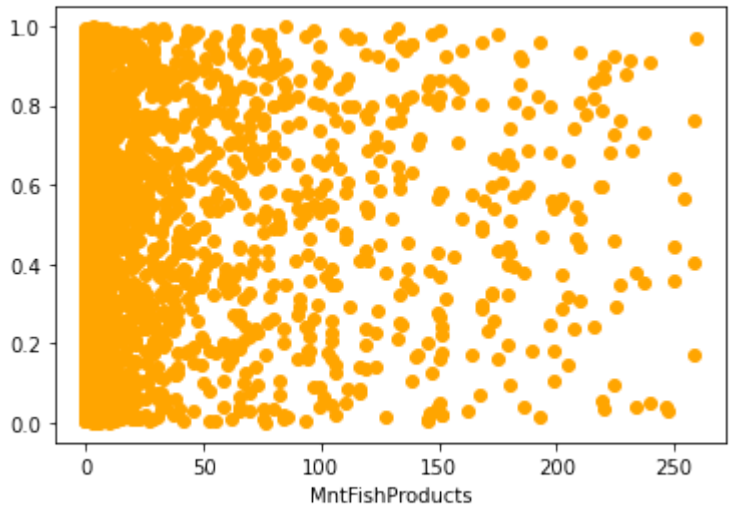

1.0
0.8
0.6
0.4
0.2
0.0
20000
40000
60000
80000
100000
Income

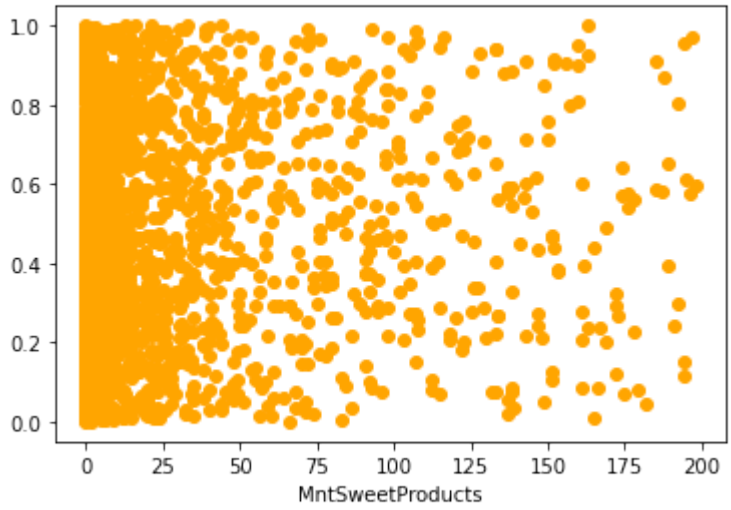

1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Recency

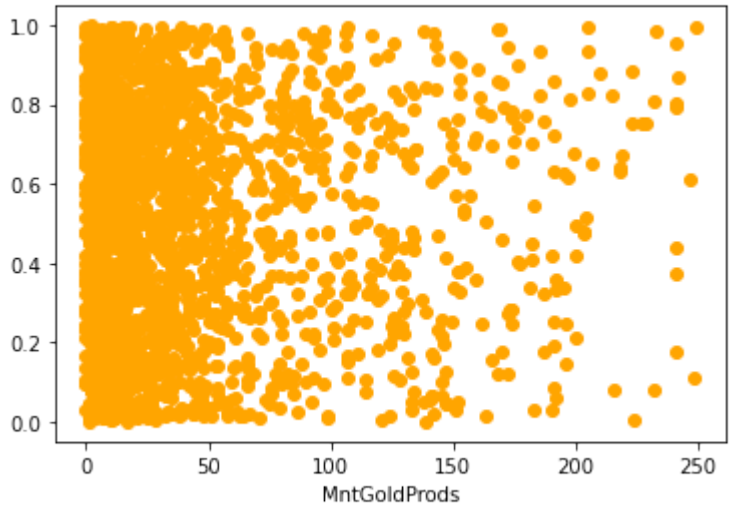

1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
MntWines

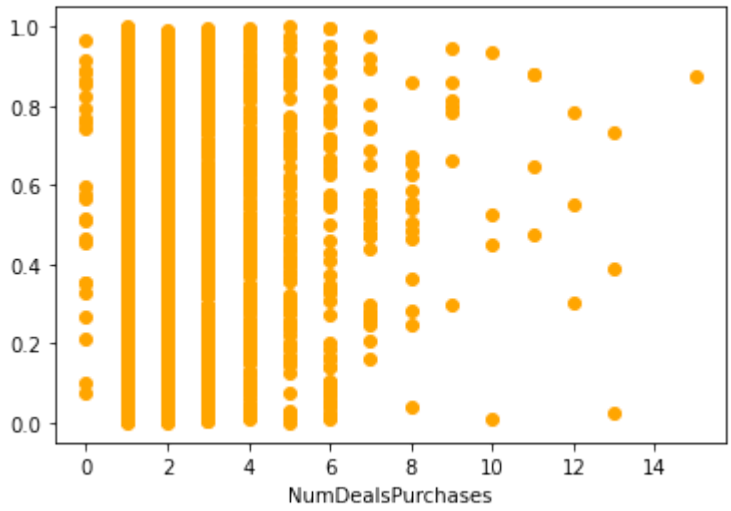

1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
MntFruits

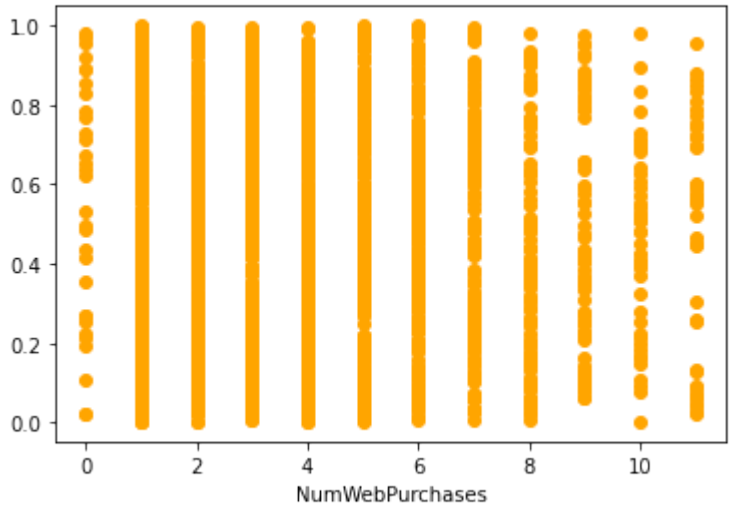

1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
MntMeatProducts

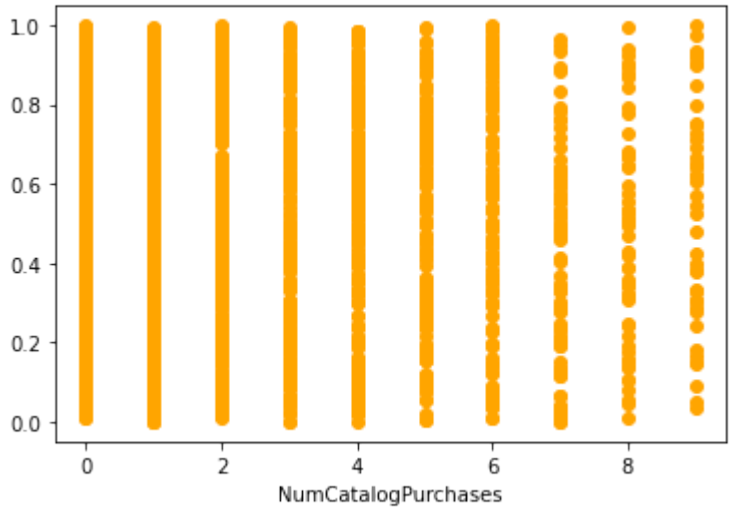

1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
MntFishProducts

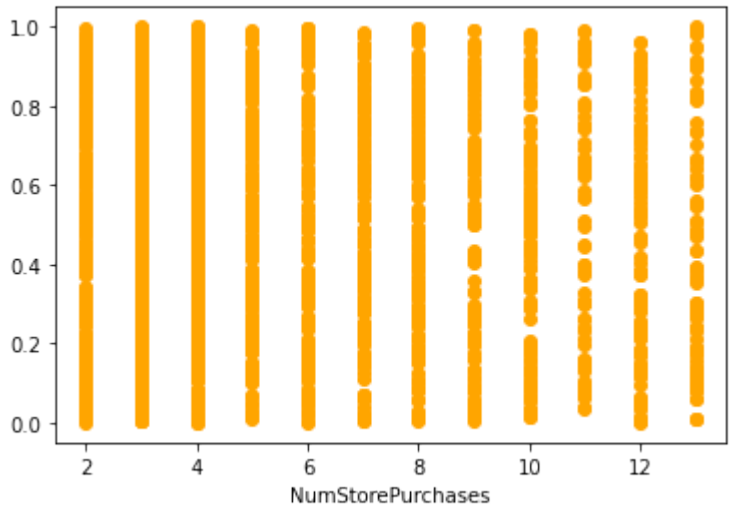

1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
MntSweetProducts

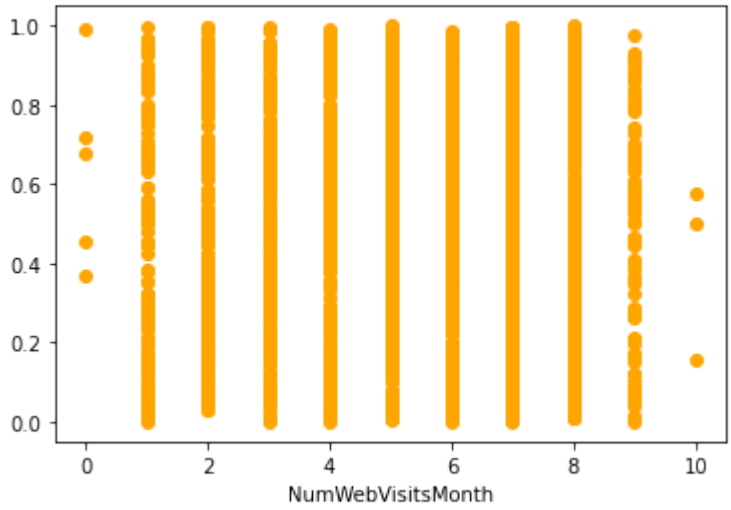

1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
MntGoldProds

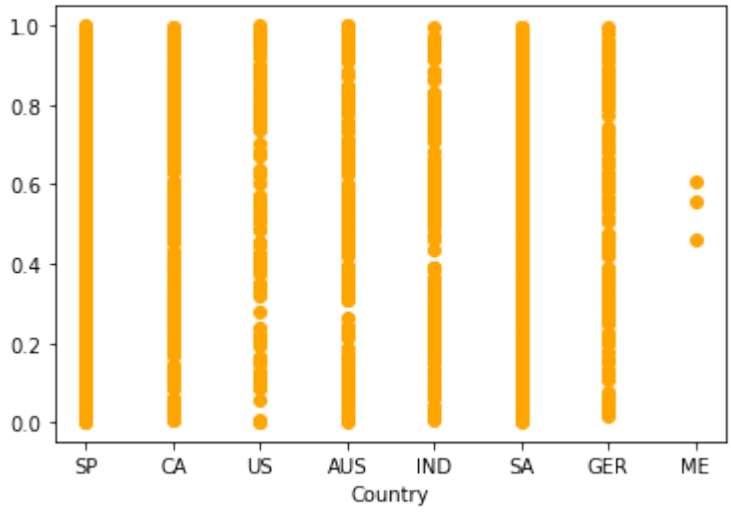

1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
NumDealsPurchases

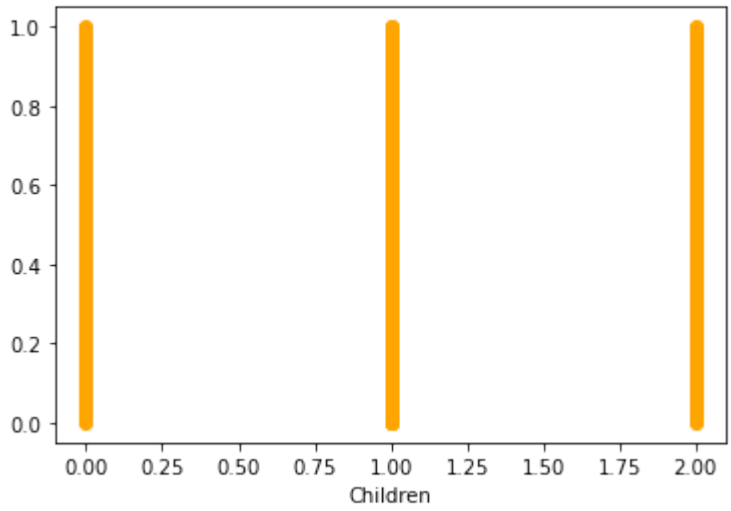

1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
NumWebPurchases

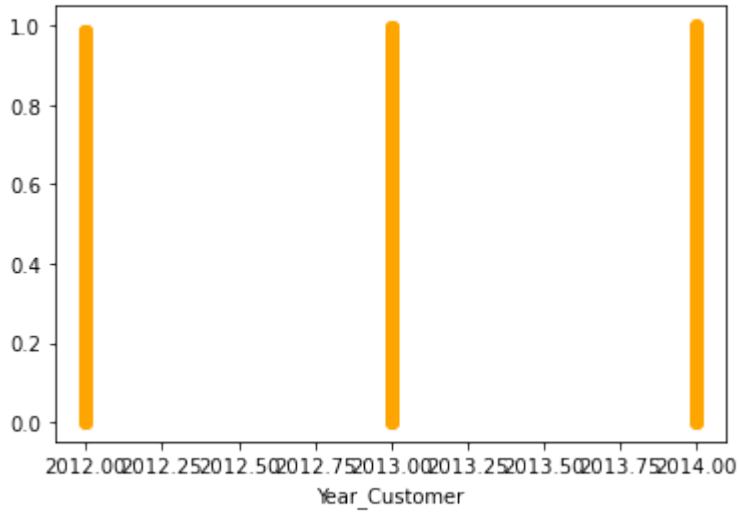

1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
NumCatalogPurchases

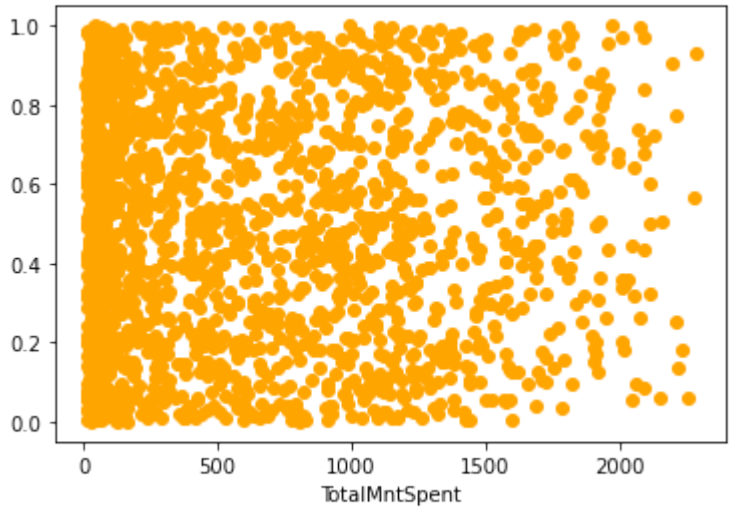

1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
12
NumStorePurchases

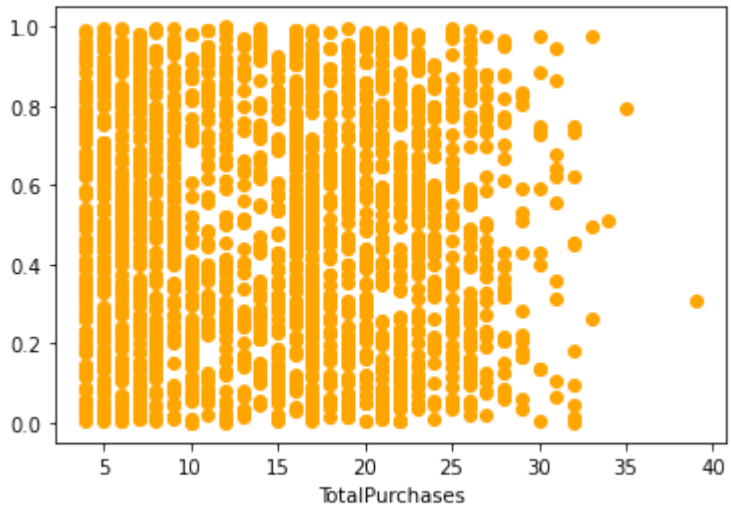

1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
NumWebVisitsMonth

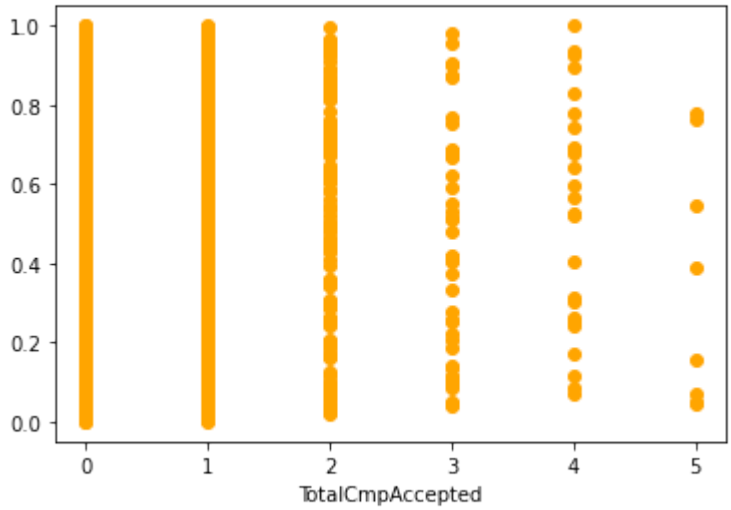

1.0
0.8
0.6
0.4
0.2
0.0
SP
CA
US
AUS
IND
SA
GER
ME
Country


1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Children


1.0
0.8
0.6
0.4
0.2
0.0
2012.00
2012.25
2012.50
2012.75
2013.00
2013.25
2013.50
2013.75
2014.00
Year_Customer


1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
TotalMntSpent




**Important Observations:**

- Year_Birth: fall bewteen 1940-1995
- Income: Between $(0-100000)
- TotalPurchases: 0-32
- Amount spent on each product category:
  - MntWines: $(0-1450)
  - MntFruits: $(0-200)
  - MntMeatProducts: $(0-1000)
  - MntFishProducts: $(0-250)
  - MntSweetProducts: $(0-200)
  - MntGoldProds: : $(0-250)
- TotalMntSpent: $(0-2300)

In [ ]:

In [218…

```
mar_df
```

Out[218…

| | Year_Birth | Education | Marital_Status | Income | Recency | MntWines | MntFruits | MntMeatProdu |
|---|---|---|---|---|---|---|---|---|
| **0** | 1970 | Graduation | Divorced | 84835.0 | 0 | 189 | 104 | 3 |
| **1** | 1961 | Graduation | Single | 57091.0 | 0 | 464 | 5 | |
| **2** | 1958 | Graduation | Married | 67267.0 | 0 | 134 | 11 | |
| **3** | 1967 | Graduation | Together | 32474.0 | 0 | 10 | 0 | |
| **4** | 1989 | Graduation | Single | 21474.0 | 0 | 6 | 16 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **2079** | 1976 | PhD | Divorced | 66476.0 | 99 | 372 | 18 | 1 |
| **2080** | 1977 | 2n Cycle | Married | 31056.0 | 99 | 5 | 10 | |
| **2081** | 1976 | Graduation | Divorced | 46310.0 | 99 | 185 | 2 | |
| **2082** | 1978 | Graduation | Married | 65819.0 | 99 | 267 | 38 | 7 |
| **2083** | 1969 | PhD | Married | 94871.0 | 99 | 169 | 24 | 5 |

2084 rows × 29 columns

## As the features are close to 30, we would first find the correlations between different features and then plot the necessary 2-D Plots

## Correlation:

### Now, let us find out the correlation between various features. Are they positively correlated, negatively correlated , or not related at all.

- We will use the Spearman Rank Correlation Coefficient (r) since it will perform well for both linearly and non-linearly related features, unlike Pearson Correlation Coefficient which will perform well only for linearly related features.

In [ ]:

In [120…

```
correlation = mar_df.corr(method='spearman')
correlation
```

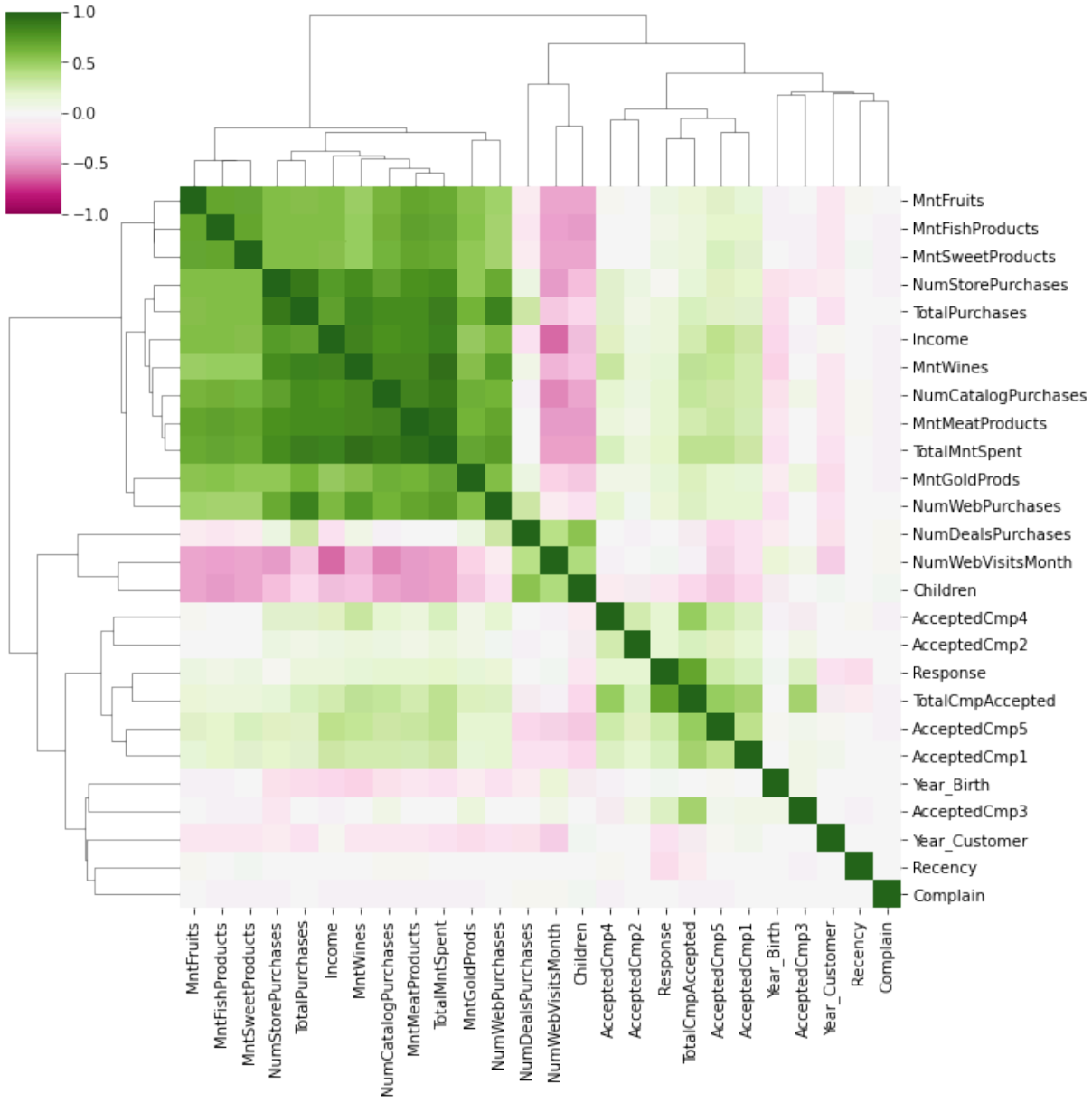
Out[120…

| | Year_Birth | Income | Recency | MntWines | MntFruits | MntMeatProducts | Mn |
|---|---|---|---|---|---|---|---|
| **Year_Birth** | 1.000000 | -0.231760 | -0.007208 | -0.247716 | -0.037959 | -0.132647 | |
| **Income** | -0.231760 | 1.000000 | 0.001514 | 0.847712 | 0.576913 | 0.824959 | |
| **Recency** | -0.007208 | 0.001514 | 1.000000 | 0.006737 | 0.024096 | 0.017044 | |
| **MntWines** | -0.247716 | 0.847712 | 0.006737 | 1.000000 | 0.498293 | 0.829724 | |
| **MntFruits** | -0.037959 | 0.576913 | 0.024096 | 0.498293 | 1.000000 | 0.707947 | |
| **MntMeatProducts** | -0.132647 | 0.824959 | 0.017044 | 0.829724 | 0.707947 | 1.000000 | |
| **MntFishProducts** | -0.036148 | 0.572464 | 0.006392 | 0.504895 | 0.697164 | 0.722368 | |
| **MntSweetProducts** | -0.008360 | 0.569434 | 0.034305 | 0.500359 | 0.688370 | 0.703149 | |

| | Year_Birth | Income | Recency | MntWines | MntFruits | MntMeatProducts | Mn |
|---|---|---|---|---|---|---|---|
| **MntGoldProds** | -0.090882 | 0.513370 | 0.010788 | 0.567399 | 0.553538 | 0.639455 | |
| **NumDealsPurchases** | -0.099664 | -0.174049 | -0.003475 | 0.070374 | -0.101742 | -0.020146 | |
| **NumWebPurchases** | -0.169804 | 0.595293 | -0.002828 | 0.750219 | 0.472663 | 0.705136 | |
| **NumCatalogPurchases** | -0.194610 | 0.795180 | 0.020061 | 0.829019 | 0.622372 | 0.846731 | |
| **NumStorePurchases** | -0.194327 | 0.754859 | 0.002457 | 0.815974 | 0.576892 | 0.796894 | |
| **NumWebVisitsMonth** | 0.147553 | -0.630379 | -0.022532 | -0.394308 | -0.440954 | -0.483677 | |
| **AcceptedCmp3** | 0.071813 | -0.043070 | -0.032886 | 0.002560 | -0.012970 | -0.020563 | |
| **AcceptedCmp4** | -0.061240 | 0.222132 | 0.016167 | 0.312571 | 0.022230 | 0.136731 | |
| **AcceptedCmp5** | 0.021934 | 0.374450 | 0.000413 | 0.347733 | 0.215475 | 0.312742 | |
| **AcceptedCmp1** | 0.003388 | 0.311045 | -0.015769 | 0.285978 | 0.155890 | 0.273239 | |
| **AcceptedCmp2** | -0.010052 | 0.104040 | -0.004458 | 0.135762 | -0.003573 | 0.062902 | |
| **Response** | 0.031438 | 0.122575 | -0.211524 | 0.160047 | 0.113533 | 0.192039 | |
| **Complain** | -0.008636 | -0.032893 | 0.000711 | -0.036307 | -0.008453 | -0.024761 | |
| **Children** | -0.082244 | -0.352934 | 0.004379 | -0.322066 | -0.451674 | -0.474348 | |
| **Year_Customer** | -0.020947 | 0.017776 | -0.012985 | -0.137692 | -0.115543 | -0.145846 | |
| **TotalMntSpent** | -0.176301 | 0.860949 | 0.010347 | 0.934235 | 0.674368 | 0.938923 | |
| **TotalPurchases** | -0.204805 | 0.725022 | 0.006070 | 0.859624 | 0.569089 | 0.823438 | |
| **TotalCmpAccepted** | -0.009304 | 0.282750 | -0.107679 | 0.355258 | 0.142075 | 0.284925 | |

In [121…

```
sns.clustermap(correlation, cmap='PiYG', vmin=-1.0, vmax=1.0, center=0);
```




**Let us plot the 2-D scatter and line plots to better understand the correlations between them**

- Income:

  - TotalMntSpent, MntWines, MntMeatProducts, NumCatalogPurchases, NumStorePurchases, TotalPurchases are positively correlated with 'Income' (r >= 0.7)

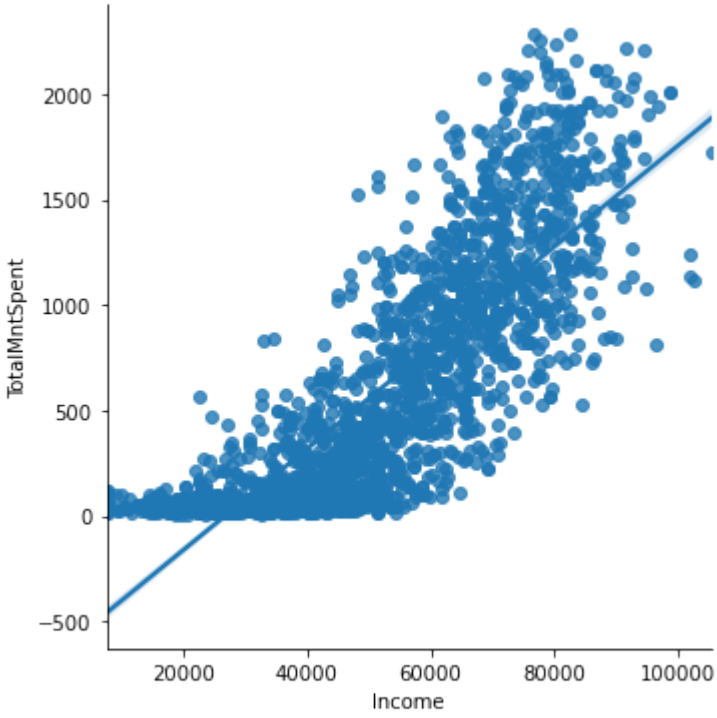
  - NumWebVisitsMonth is negatively correlated with 'Income' (r = -0.62)

In [122…

```
sns.lmplot(x='Income', y='TotalMntSpent', data=mar_df)
```

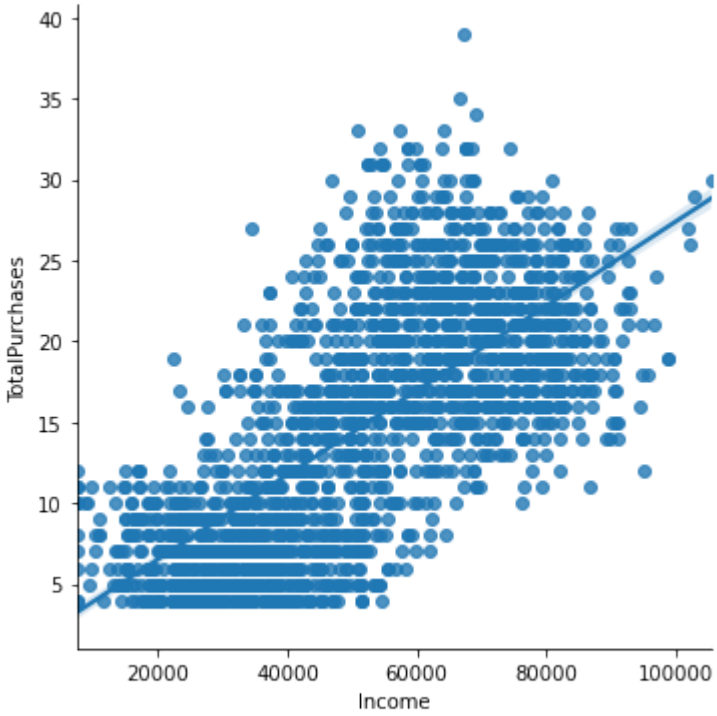
Out[122…

```
<seaborn.axisgrid.FacetGrid at 0x1fc04400490>
```



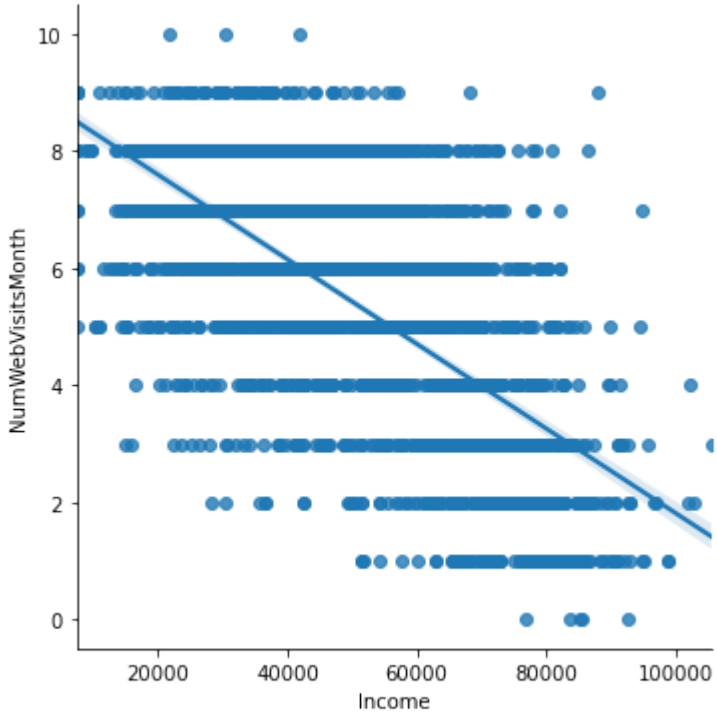

In [123…
```
sns.lmplot(x='Income', y='TotalPurchases', data=mar_df)
```

Out[123… <seaborn.axisgrid.FacetGrid at 0x1fc04182b80>




In [124…
```
sns.lmplot(x='Income', y='NumWebVisitsMonth', data=mar_df)
```

Out[124… <seaborn.axisgrid.FacetGrid at 0x1fc047c4160>




In [ ]:

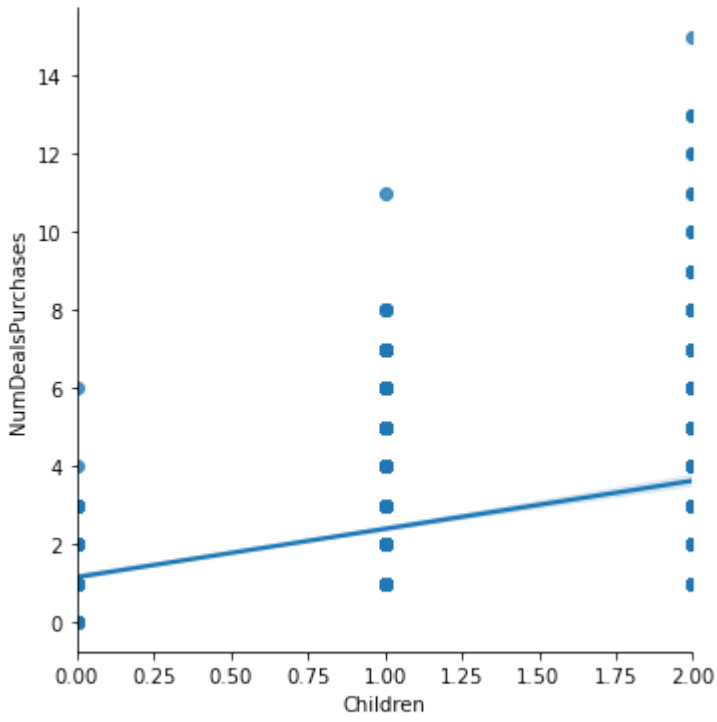
- Children:

  - NumDealsPurchases is positively correlated with 'Children'

  - TotalMntSpent, MntMeatProducts, NumCatalogPurchases, MntFishProducts are negatively correlated with 'Children'

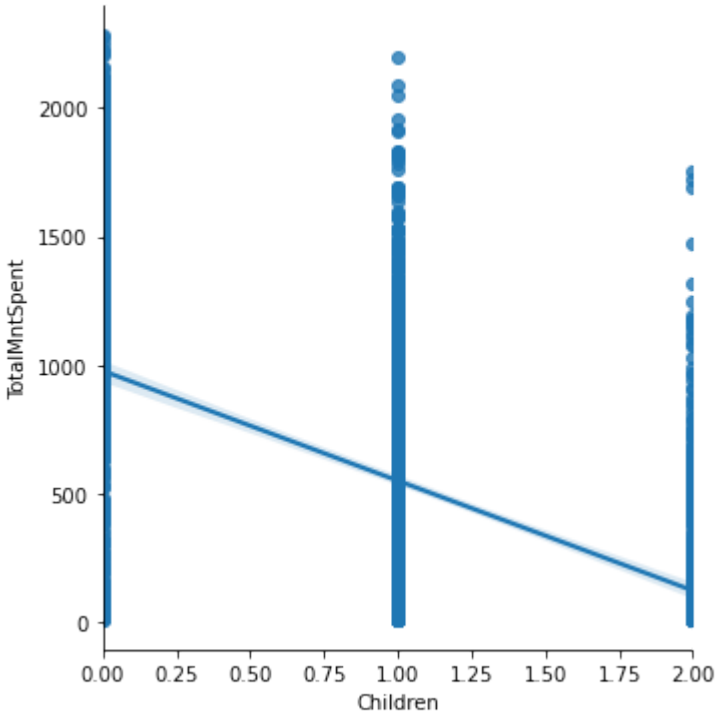
In [125…
```
sns.lmplot(x='Children', y='NumDealsPurchases', data=mar_df)
```

Out[125… <seaborn.axisgrid.FacetGrid at 0x1fc04727a30>



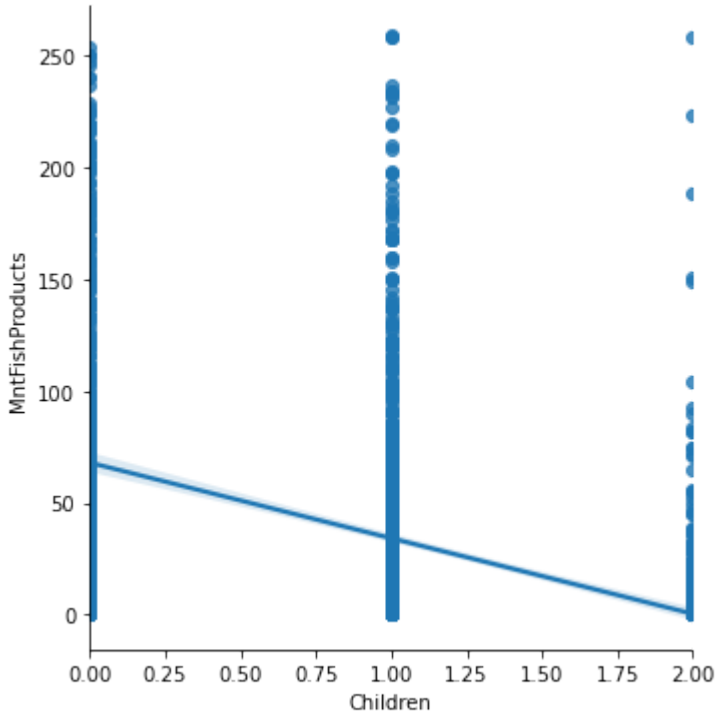
In [126…
```
sns.lmplot(x='Children', y='TotalMntSpent', data=mar_df)
```

Out[126…
```
<seaborn.axisgrid.FacetGrid at 0x1fc043a2520>
```



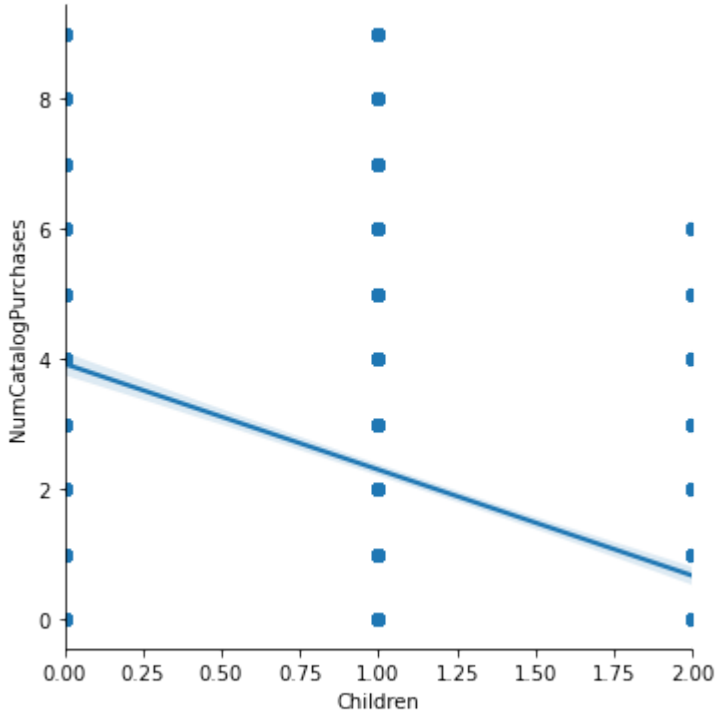

In [127…
```
sns.lmplot(x='Children', y='MntFishProducts', data=mar_df)
```

Out[127…
```
<seaborn.axisgrid.FacetGrid at 0x1fc043f7c40>
```




In [128…
```
sns.lmplot(x='Children', y='NumCatalogPurchases', data=mar_df)
```

Out[128…
```
<seaborn.axisgrid.FacetGrid at 0x1fc0467eee0>
```




In [ ]:

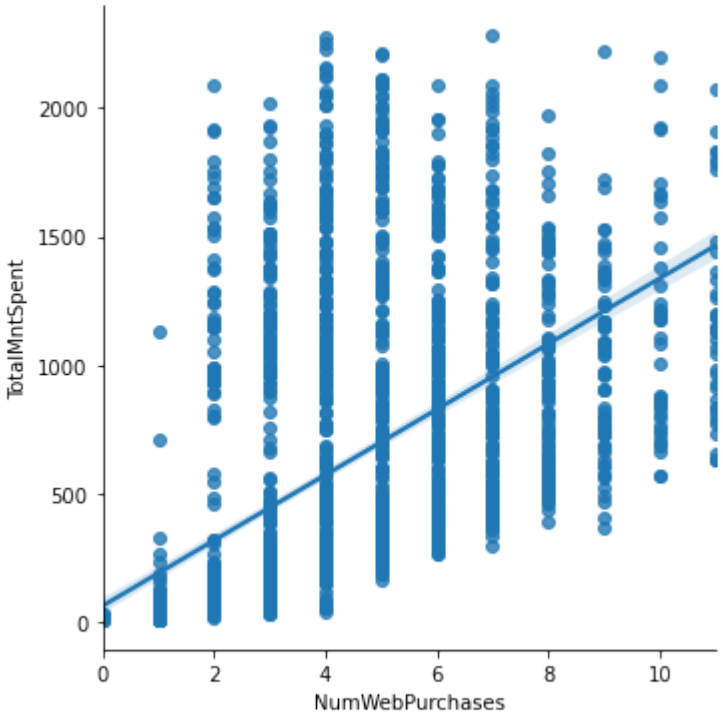
- TotalMntSpent:

  - NumWebPurchases, NumStorePurchases, NumCatalogPurchases is positively correlated with 'TotalMntSpent'

  - NumWebVisitsMonth, Children are negatively correlated with 'TotalMntSpent'

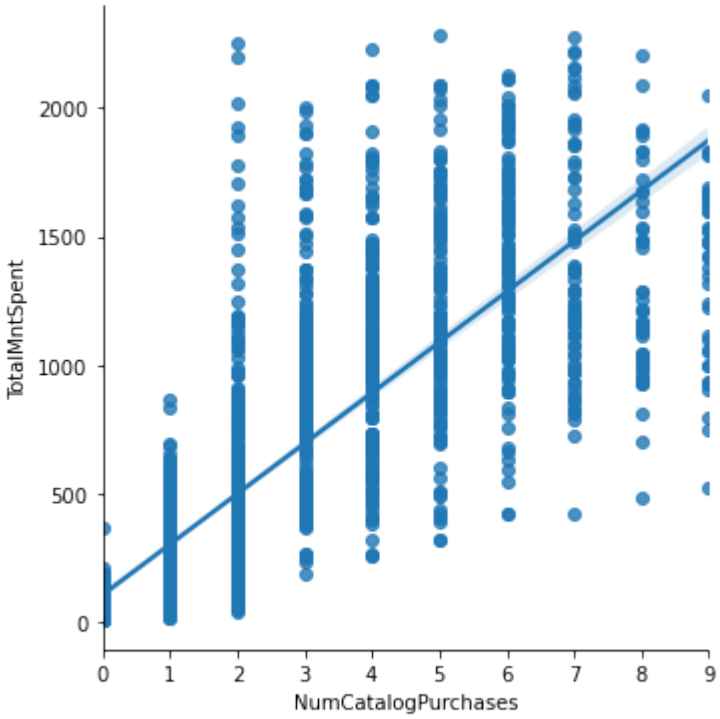
In [129…
```
sns.lmplot(x='NumWebPurchases', y='TotalMntSpent', data=mar_df)
```

Out[129… <seaborn.axisgrid.FacetGrid at 0x1fc040dff10>



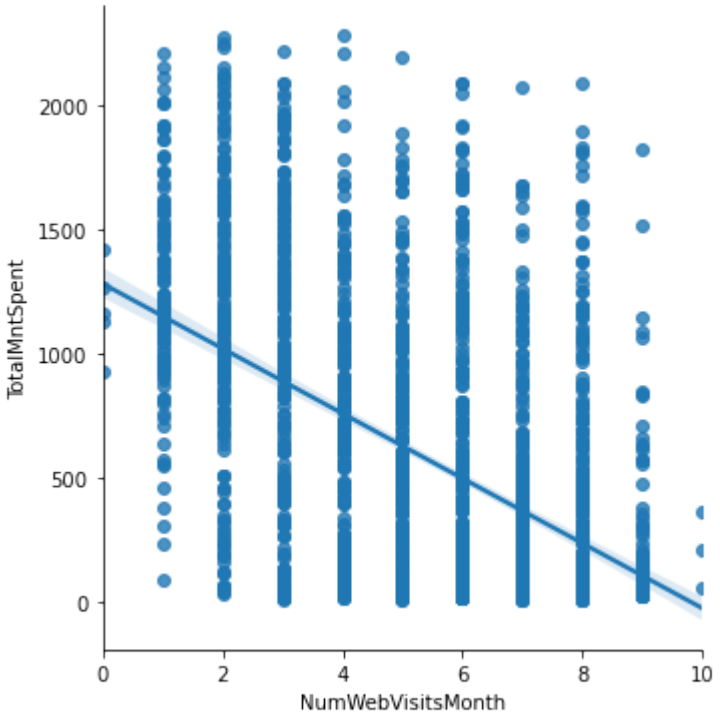
In [130…
```
sns.lmplot(x='NumCatalogPurchases', y='TotalMntSpent', data=mar_df)
```

Out[130…
```
<seaborn.axisgrid.FacetGrid at 0x1fc044ae070>
```



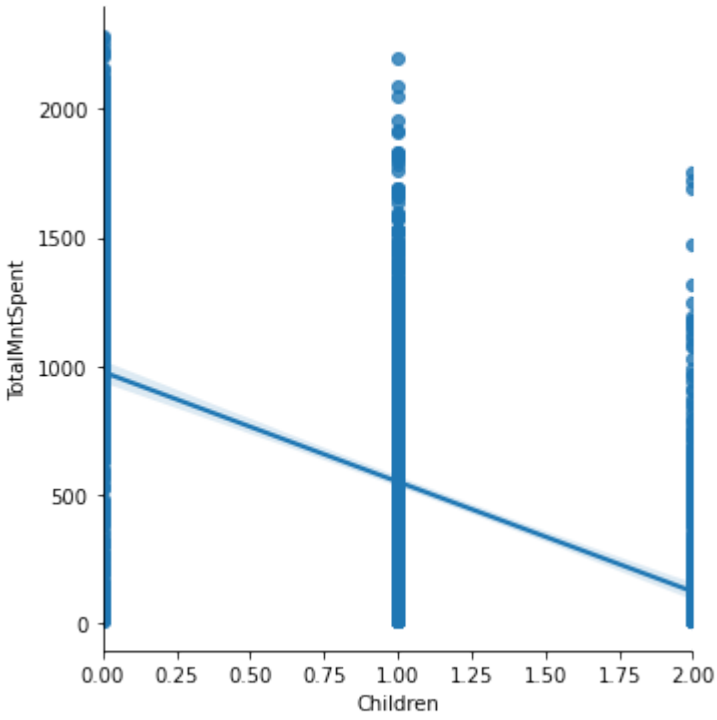

In [131…
```
sns.lmplot(x='NumWebVisitsMonth', y='TotalMntSpent', data=mar_df)
```

Out[131…
```
<seaborn.axisgrid.FacetGrid at 0x1fc04d81c70>
```




In [132…
```
sns.lmplot(x='Children', y='TotalMntSpent', data=mar_df)
```

Out[132…
```
<seaborn.axisgrid.FacetGrid at 0x1fc04d76df0>
```




In [ ]:

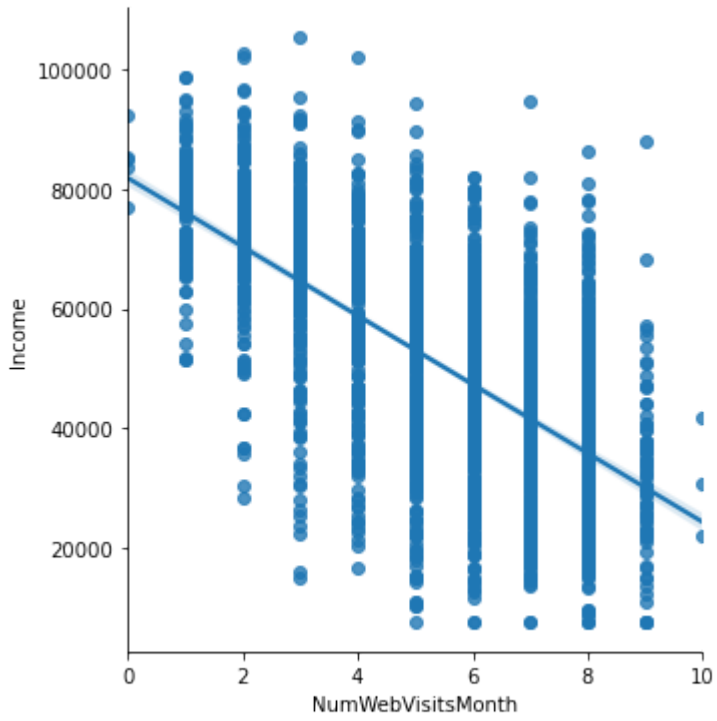
- NumWebVisitsMonth:

  - Children, NumDealsPurchases are positively correlated with 'NumWebVisitsMonth'

  - Income, NumCatalogPurchases, TotalMntSpent are negatively correlated with 'NumWebVisitsMonth'

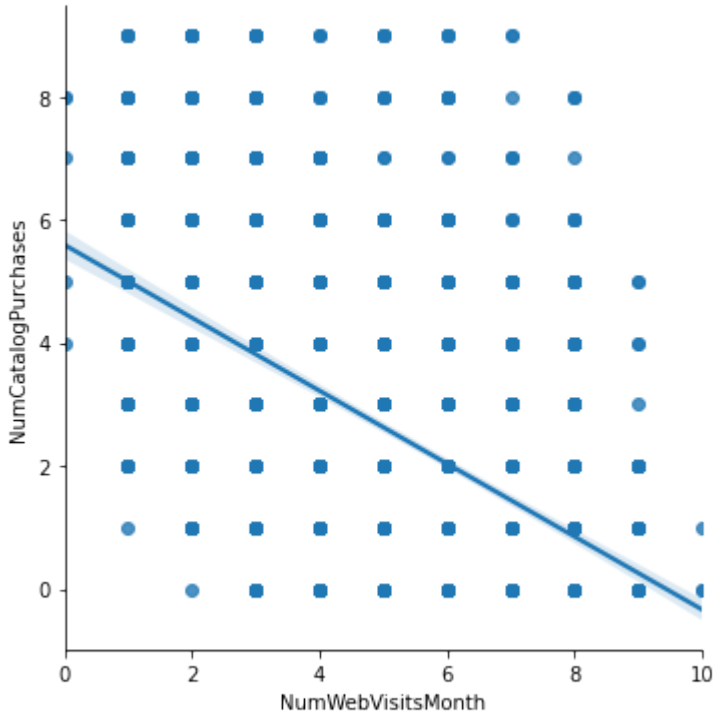
In [133…
```
sns.lmplot(x='NumWebVisitsMonth', y='Income', data=mar_df)
```

Out[133…
```
<seaborn.axisgrid.FacetGrid at 0x1fc04dd6d30>
```



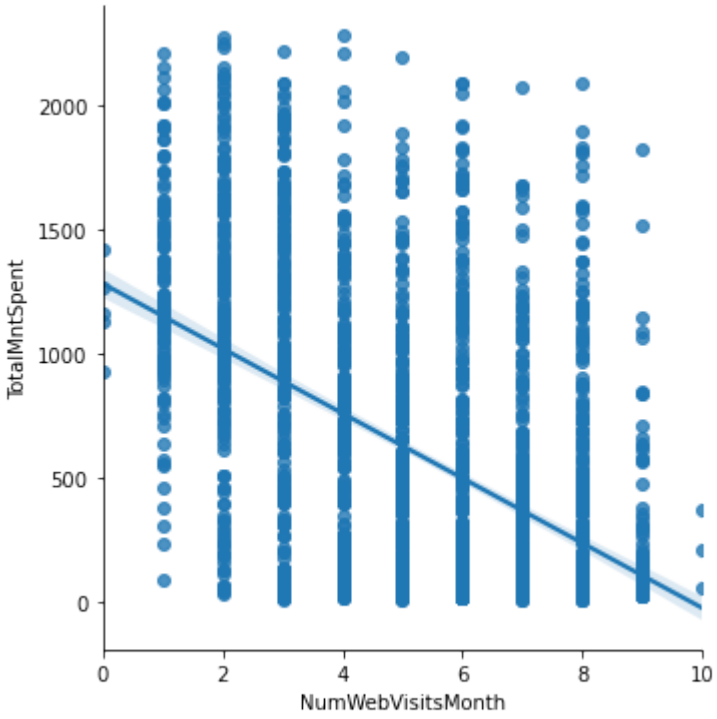
In [134…
```
sns.lmplot(x='NumWebVisitsMonth', y='NumCatalogPurchases', data=mar_df)
```

Out[134…
```
<seaborn.axisgrid.FacetGrid at 0x1fc04e9ba00>
```



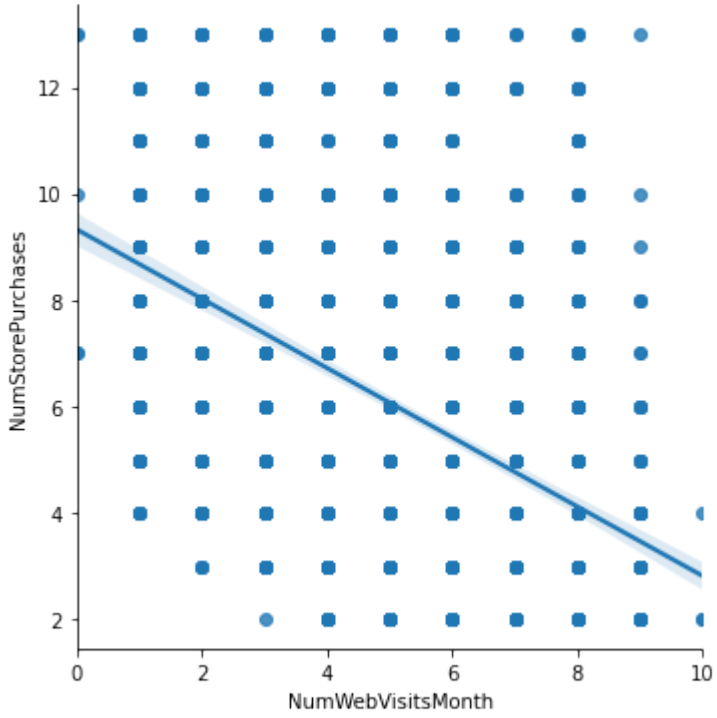

In [135…
```
sns.lmplot(x='NumWebVisitsMonth', y='TotalMntSpent', data=mar_df)
```

Out[135…
```
<seaborn.axisgrid.FacetGrid at 0x1fc04e7ed30>
```



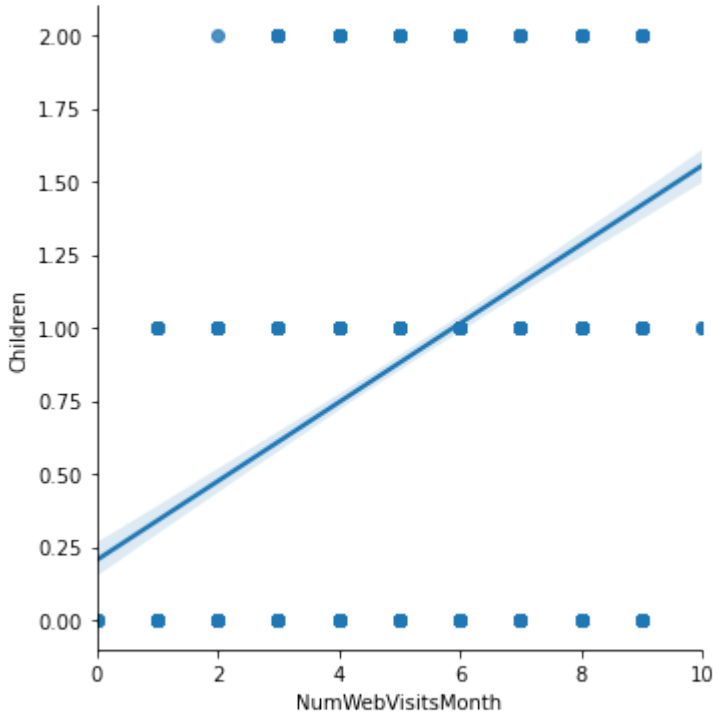

In [220…
```
sns.lmplot(x='NumWebVisitsMonth', y='NumStorePurchases', data=mar_df)
```

Out[220…
```
<seaborn.axisgrid.FacetGrid at 0x1fc08c09bb0>
```



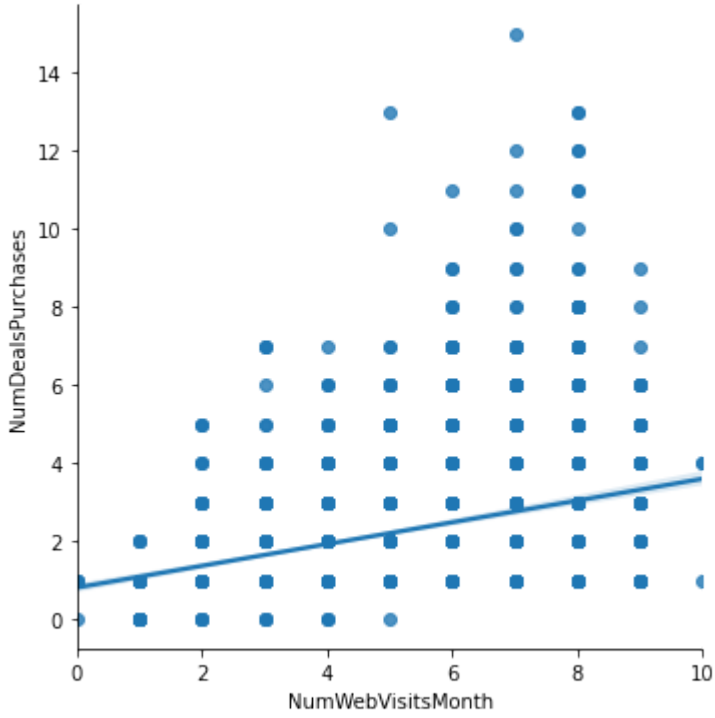

In [136…
```
sns.lmplot(x='NumWebVisitsMonth', y='Children', data=mar_df)
```

Out[136… <seaborn.axisgrid.FacetGrid at 0x1fc04f4ce80>




In [137…
```
sns.lmplot(x='NumWebVisitsMonth', y='NumDealsPurchases', data=mar_df)
```

Out[137… <seaborn.axisgrid.FacetGrid at 0x1fc04f2cc40>



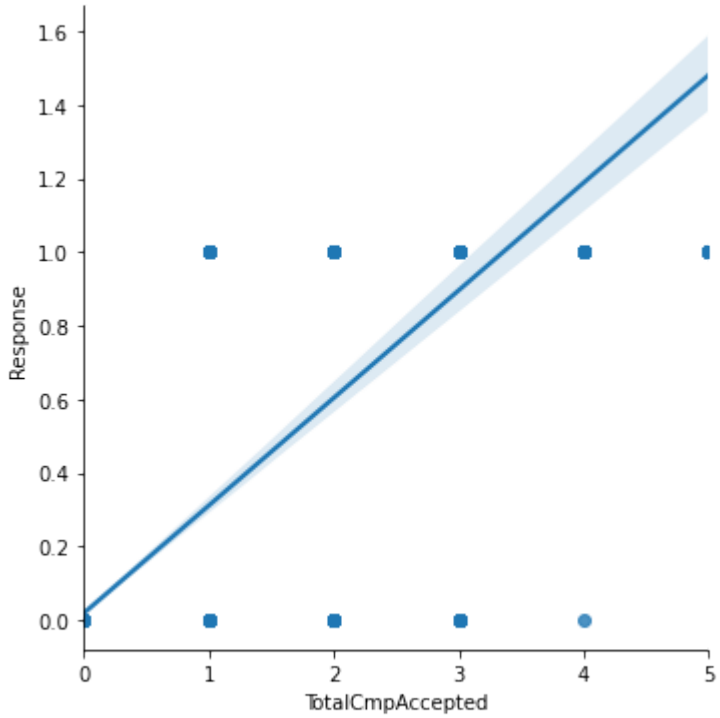

In [ ]:

- TotalCmpAccepted:

  - Response is positively correlated with TotalCmpAccepted

In [138…
```
sns.lmplot(x='TotalCmpAccepted', y='Response', data=mar_df)
```

Out[138… <seaborn.axisgrid.FacetGrid at 0x1fc0714fa30>




In [ ]:

# Now, let us answer some important data analysis questions

### Q) What is the total amount spent on each product? What are the best selling and least selling products' by amount spent?

In [143…
```
mnt_products = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
                'MntSweetProducts', 'MntGoldProds']
```

In [144…
```
print('Total amount spent on each item (in $):\n')

for product in mnt_products:
    amnt = mar_df[product].sum()
    print(f'{product}: {amnt}')
```

```
Total amount spent on each item (in $):

MntWines: 615819
MntFruits: 53325
MntMeatProducts: 324634
MntFishProducts: 76408
MntSweetProducts: 55164
MntGoldProds: 88801
```

- Best selling product: Wines with $549781
- Least selling product: Fruits with $49642

### Q) People from which country spent the most amount? And the least amount?

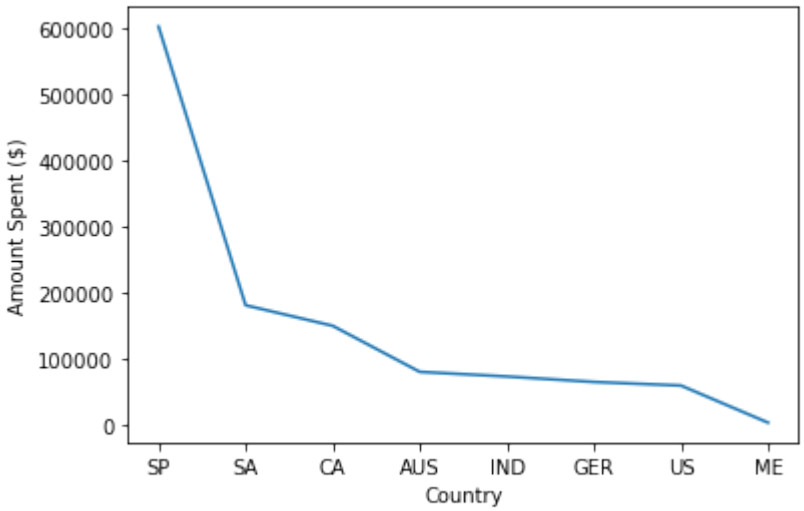
In [145…
```
mar_df.groupby('Country')['TotalMntSpent'].sum().sort_values(ascending=False)
```

Out[145…
```
Country
SP     603111
SA     180920
CA     149824
AUS     80052
IND     73030
GER     64761
US      59331
ME       3122
Name: TotalMntSpent, dtype: int64
```

- People from Spain (SP) spent the most amount with $540175
- People from Mexico (ME) spent the least amount with $3122

In [146…
```
mar_df.groupby('Country')['TotalMntSpent'].sum().sort_values(ascending=False).plot()
plt.ylabel('Amount Spent ($)')
```

Out[146…
```
Text(0, 0.5, 'Amount Spent ($)')
```




In [ ]:

**Q3) What are the total number of purchases from each country?**

In [147…
```
mar_df.groupby('Country')['TotalPurchases'].sum().sort_values(ascending=False)
```

Out[147…
```
Country
SP     14905
SA      4564
CA      3778
AUS     2081
IND     1979
GER     1567
US      1565
ME        59
Name: TotalPurchases, dtype: int64
```

- Spain has the most item purchases with 14060 while Mexico with the least at 59 item purchases

**Q) What is the average income of the customer in each country?**

In [148…
```
mar_df.groupby('Country')['Income'].mean().sort_values(ascending=False)
```

Out[148…
```
Country
ME     57680.333333
US     52709.485149
CA     52111.016194
SA     51588.581699
GER    51279.346847
AUS    51184.564626
SP     50730.332849
IND    48745.094891
Name: Income, dtype: float64
```

Mexico has the highest average income (57680) while India has the least average income (48745)

**Q) What is the amount spent by each group based on their education level? Did PhD customers spent more than Graduate customers?**

In [149…
```
mar_df.groupby('Education')['TotalMntSpent'].sum().sort_values(ascending=False)
```

Out[149…
```
Education
Graduation    628299
```

```
PhD           286118
Master        203655
2n Cycle       91662
Basic           4417
Name: TotalMntSpent, dtype: int64
```

Customers' with a Graduation degree spent the most amount followed by PhD Customers'.

Customers' with basic degree spent the least amount.

**Q) Which group spent the most amount wrt marital status?**

In [150…
```
mar_df.groupby('Marital_Status')['TotalMntSpent'].sum().sort_values(ascending=False)
```

Out[150…
```
Marital_Status
Married     452870
Together    318319
Single      256930
Divorced    132154
Widow        51091
Absurd        1169
YOLO           848
Alone          770
Name: TotalMntSpent, dtype: int64
```

Married group spent the most amount

**Q) Are people with no children made the most purchases than people with more children?**

In [151…
```
mar_df.groupby('Children')['TotalPurchases'].sum().sort_values(ascending=False)
```

Out[151…
```
Children
1    15717
0     9712
2     5069
Name: TotalPurchases, dtype: int64
```

No. Customers' with 1 child made the most purchases followed by no children

**Q) What are the numbers of purchases made?**

In [152…
```
purchases = ['NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStor
```

In [153…
```
for purchase in purchases:
    amnt = mar_df[purchase].sum()
    print(f'{purchase}: {amnt}')
```

```
NumDealsPurchases: 4801
NumWebPurchases: 8481
NumCatalogPurchases: 5041
NumStorePurchases: 12175
```

- Store purchases are more than web purchases

**Q) What is the total traffic for the website?**

In [154…
```
# Total number of website visits/traffic
mar_df['NumWebVisitsMonth'].sum()
```

Out[154…
```
11152
```

**Q) What is the website traffic from each country? Which country customers most visited the website?**

In [155…
```
mar_df.groupby('Country')['NumWebVisitsMonth'].sum().sort_values(ascending=False)
```

Out[155…
```
Country
SP     5503
SA     1648
CA     1305
AUS     775
IND     759
GER     589
US      555
ME       18
Name: NumWebVisitsMonth, dtype: int64
```

- Maximum traffic is from SP (Spain) and minimum from ME (Mexico)

**Q) Which is the most successful marketing campaign based on the number of acceptances?**

In [156…
```
campaigns = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'Accept
             'Response']
```

In [157…
```
for campn in campaigns:
    score = mar_df[campn].sum()
    print(f'{campn}: {score}')
```

```
AcceptedCmp1: 118
AcceptedCmp2: 28
AcceptedCmp3: 143
AcceptedCmp4: 154
AcceptedCmp5: 136
Response: 294
```

Most successful campaign is 'Response' by number of acceptances

**Q) Which country has the most number of accepted campaigns?**

In [158…
```
mar_df.groupby('Country')['TotalCmpAccepted'].sum().sort_values(ascending=False)
```

Out[158…
```
Country
SP     467
SA     118
CA     103
GER     50
IND     48
AUS     47
US      37
ME       3
Name: TotalCmpAccepted, dtype: int64
```

Spain (SP) has the most number of accepted campaigns with 467

In [221…
```
mar_df.groupby('Country')['Response'].sum().sort_values(ascending=False)
```

Out[221…
```
Country
SP     157
SA      46
CA      31
AUS     19
GER     15
IND     12
US      12
```

```
ME       2
Name: Response, dtype: int64
```

- The most successful campaign 'Response' has most number of acceptances from SP and least from ME

In [ ]: