Market Analytics

by Yueh-Han Chen

Context

> I'm a marketing data analyst and I've been told by the Chief Marketing Officer that recent marketing campaigns have not been as effective as they were expected to be. I need to analyze the data set to understand this problem and propose data-driven solutions.
>
> Kaggle link: https://www.kaggle.com/jackdaoud/marketing-data

Section 01: Exploratory Data Analysis

- Are there any null values or outliers? How will you wrangle/handle them?
- Are there any useful variables that you can engineer with the given data?
- Do you notice any patterns or anomalies in the data? Can you plot them?

Section 02: Statistical Analysis

Please run statistical tests in the form of regressions to answer these questions & propose data-driven action recommendations to your CMO. Make sure to interpret your results with non-statistical jargon so your CMO can understand your findings.

- What factors are significantly related to the number of store purchases?
- Your supervisor insists that people who buy gold are more conservative. Therefore, people who spent an above average amount on gold in the last 2 years would have more in store purchases. Justify or refute this statement using an appropriate statistical test
- Fish has Omega 3 fatty acids which are good for the brain. Accordingly, do "Married PhD candidates" have a significant relation with amount spent on fish?

Section 03: Data Visualization

Please plot and visualize the answers to the below questions.

- Which marketing campaign is most successful?

- What does the average customer look like for this company? Which products are performing best?
- Which channels are underperforming?

Section 04: CMO Recommendations

- Bring together everything from Sections 01 to 03 and provide data-driven recommendations/suggestions to your CMO.

Gather

In [1]:
```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline

# suppress warnings from final output
import warnings
warnings.simplefilter("ignore")

# set up to view all the info of the columns
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [2]:
```
#load data
df = pd.read_csv('marketing_data.csv')
```

Features Information from Kaggle:

- ID: Customer's unique identifier
- Year_Birth: Customer's birth year
- Education: Customer's education level
- Marital_Status: Customer's marital status
- Income: Customer's yearly household income
- Kidhome: Number of children in customer's household
- Tennhome: Number of teenagers in customer's household
- Dt_Customer: Date of customer's enrollment with the company
- Recency: Number of days since customer's last purchase
- MntWines: Amount spent on wine in the last 2 years
- MntFruits: Amount spent on fruits in the last 2 years
- MntMeatProducts: Amount spent on meat in the last 2 years
- MntFishProducts: Amount spent on fish in the last 2 years
- MntSweetProducts: Amount spent on sweets in the last 2 years
- MntGoldProds: Amount spent on gold in the last 2 years
- NumDealsPurchase: Number of purchases made with a discount
- NumWebPurchase: Number of purchases made through the company's web site
- NumCatalogPurchase: Number of purchases made using a catalogue
- NumStorePurchase: Number of purchases made directly in stores
- NumWebVisitsMonth: Number of visits to company's web site in the last month
- AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
- AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise
- AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise
- AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise
- AcceptedCmp2: 2 if customer accepted the offer in the 1st campaign, 0 otherwise
- Respones: 1 if customer accepted the offer in the last campaign, 0 otherwise
- Complain: 1 if customer complained in the last 2 years, 0 otherwise
- Country: Customer's location

Assess

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:

```
def basic_info(df):
    print("This dataset has ", df.shape[1], " columns and ", df.shape[0], " r
    print("This dataset has ", df[df.duplicated()].shape[0], " duplicated row
    print(" ")
    print("Descriptive statistics of the numeric features in the dataset: ")
    print(" ")
    print(df.describe())
    print(" ")
    print("Information about this dataset: ")
    print(" ")
    print(df.info())
```

In [5]:

```
basic_info(df)
```

```
This dataset has  28  columns and  2240  rows.
This dataset has  0  duplicated rows.

Descriptive statistics of the numeric features in the dataset:

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

          MntWines    MntFruits  MntMeatProducts  MntFishProducts  \
count  2240.000000  2240.000000      2240.000000      2240.000000
mean    303.935714    26.302232       166.950000        37.525446
std     336.597393    39.773434       225.715373        54.628979
min       0.000000     0.000000         0.000000         0.000000
25%      23.750000     1.000000        16.000000         3.000000
50%     173.500000     8.000000        67.000000        12.000000
75%     504.250000    33.000000       232.000000        50.000000
max    1493.000000   199.000000      1725.000000       259.000000
```

```
       MntSweetProducts  MntGoldProds  NumDealsPurchases  NumWebPurchases  \
count       2240.000000   2240.000000        2240.000000      2240.000000
mean          27.062946     44.021875           2.325000         4.084821
std           41.280498     52.167439           1.932238         2.778714
min            0.000000      0.000000           0.000000         0.000000
25%            1.000000      9.000000           1.000000         2.000000
50%            8.000000     24.000000           2.000000         4.000000
75%           33.000000     56.000000           3.000000         6.000000
max          263.000000    362.000000          15.000000        27.000000

       NumCatalogPurchases  NumStorePurchases  NumWebVisitsMonth  \
count          2240.000000        2240.000000        2240.000000
mean              2.662054           5.790179           5.316518
std               2.923101           3.250958           2.426645
min               0.000000           0.000000           0.000000
25%               0.000000           3.000000           3.000000
50%               2.000000           5.000000           6.000000
75%               4.000000           8.000000           7.000000
max              28.000000          13.000000          20.000000

       AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  \
count   2240.000000   2240.000000   2240.000000   2240.000000   2240.000000
mean       0.072768      0.074554      0.072768      0.064286      0.013393
std        0.259813      0.262728      0.259813      0.245316      0.114976
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

          Response     Complain
count  2240.000000  2240.000000
mean      0.149107     0.009375
std       0.356274     0.096391
min       0.000000     0.000000
25%       0.000000     0.000000
50%       0.000000     0.000000
75%       0.000000     0.000000
max       1.000000     1.000000

Information about this dataset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 columns):
ID                     2240 non-null int64
Year_Birth             2240 non-null int64
Education              2240 non-null object
Marital_Status         2240 non-null object
 Income                2216 non-null object
Kidhome                2240 non-null int64
Teenhome               2240 non-null int64
Dt_Customer            2240 non-null object
```

```
Recency                  2240 non-null int64
MntWines                 2240 non-null int64
MntFruits                2240 non-null int64
MntMeatProducts          2240 non-null int64
MntFishProducts          2240 non-null int64
MntSweetProducts         2240 non-null int64
MntGoldProds             2240 non-null int64
NumDealsPurchases        2240 non-null int64
NumWebPurchases          2240 non-null int64
NumCatalogPurchases      2240 non-null int64
NumStorePurchases        2240 non-null int64
NumWebVisitsMonth        2240 non-null int64
AcceptedCmp3             2240 non-null int64
AcceptedCmp4             2240 non-null int64
AcceptedCmp5             2240 non-null int64
AcceptedCmp1             2240 non-null int64
AcceptedCmp2             2240 non-null int64
Response                 2240 non-null int64
Complain                 2240 non-null int64
Country                  2240 non-null object
dtypes: int64(23), object(5)
memory usage: 490.1+ KB
None
```

Assessment report:

Quality issues

- There is a space in front of the income's column name
- There are dollar signs is the values of Income column
- The "Income" column has 23 missing values
- Income's type is string
- Dt_Customer's type is string

Data Cleaning

In [6]:
```
df_copy = df.copy()
```

Issue 1: There is a space in front of the income's column name

Code

In [7]:
```
df_copy.rename(columns={' Income ':'Income'}, inplace=True)
```

Test

In [8]:
```
df_copy.columns
```

Out[8]:
```
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')
```

Issue 2: There are dollar signs, spaces, commas, and dots is the values of Income column

Code

In [9]:
```
df_copy.Income = df_copy.Income.str.strip('$')
df_copy.Income = df_copy.Income.str.replace(".", "")
df_copy.Income = df_copy.Income.str.replace(",", "")
df_copy.Income = df_copy.Income.str.replace("00 ", "")
```

Test

In [10]:
```
df_copy.Income.sample(5)
```

Out[10]:
```
1103    38547
2178    86857
1721     6835
1612    14515
660     50437
Name: Income, dtype: object
```

In [11]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 columns):
ID                     2240 non-null int64
Year_Birth             2240 non-null int64
Education              2240 non-null object
Marital_Status         2240 non-null object
Income                 2216 non-null object
Kidhome                2240 non-null int64
Teenhome               2240 non-null int64
Dt_Customer            2240 non-null object
Recency                2240 non-null int64
MntWines               2240 non-null int64
MntFruits              2240 non-null int64
MntMeatProducts        2240 non-null int64
MntFishProducts        2240 non-null int64
MntSweetProducts       2240 non-null int64
MntGoldProds           2240 non-null int64
NumDealsPurchases      2240 non-null int64
NumWebPurchases        2240 non-null int64
NumCatalogPurchases    2240 non-null int64
NumStorePurchases      2240 non-null int64
NumWebVisitsMonth      2240 non-null int64
AcceptedCmp3           2240 non-null int64
AcceptedCmp4           2240 non-null int64
AcceptedCmp5           2240 non-null int64
AcceptedCmp1           2240 non-null int64
AcceptedCmp2           2240 non-null int64
Response               2240 non-null int64
Complain               2240 non-null int64
Country                2240 non-null object
dtypes: int64(23), object(5)
memory usage: 490.1+ KB
```

Issue 3: The "Income" column has 23 missing values

Issue 4: Income's type is string

Code

In [12]:

```
# divide the data into two dataframes: one has income values, and the other d
have_income = df_copy[df_copy.Income.isnull()==False]
missing_income = df_copy[df_copy.Income.isnull()==True]
```

In [13]:
```
# Convert the one that has income to int type
have_income.Income = have_income.Income.astype(int)

# give a string value of "0" to missing value, then we can convert it into in
missing_income.Income = str(have_income.Income.median())

missing_income.Income = missing_income.Income.str.replace(".5", "")
missing_income.Income = missing_income.Income.astype(int)
```

In [14]:
```
#combine the data
df_copy = missing_income.append(have_income)
```

Test

In [15]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2240 entries, 134 to 2239
Data columns (total 28 columns):
ID                     2240 non-null int64
Year_Birth             2240 non-null int64
Education              2240 non-null object
Marital_Status         2240 non-null object
Income                 2240 non-null int64
Kidhome                2240 non-null int64
Teenhome               2240 non-null int64
Dt_Customer            2240 non-null object
Recency                2240 non-null int64
MntWines               2240 non-null int64
MntFruits              2240 non-null int64
MntMeatProducts        2240 non-null int64
MntFishProducts        2240 non-null int64
MntSweetProducts       2240 non-null int64
MntGoldProds           2240 non-null int64
NumDealsPurchases      2240 non-null int64
NumWebPurchases        2240 non-null int64
NumCatalogPurchases    2240 non-null int64
NumStorePurchases      2240 non-null int64
NumWebVisitsMonth      2240 non-null int64
AcceptedCmp3           2240 non-null int64
AcceptedCmp4           2240 non-null int64
AcceptedCmp5           2240 non-null int64
AcceptedCmp1           2240 non-null int64
AcceptedCmp2           2240 non-null int64
Response               2240 non-null int64
Complain               2240 non-null int64
Country                2240 non-null object
dtypes: int64(24), object(4)
memory usage: 507.5+ KB
```

Issue 5: Dt_Customer's type is string

Code

In [16]:
```
df_copy.Dt_Customer = pd.to_datetime(df_copy.Dt_Customer)
```

Test

In [17]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2240 entries, 134 to 2239
Data columns (total 28 columns):
ID                     2240 non-null int64
Year_Birth             2240 non-null int64
Education              2240 non-null object
Marital_Status         2240 non-null object
Income                 2240 non-null int64
Kidhome                2240 non-null int64
Teenhome               2240 non-null int64
Dt_Customer            2240 non-null datetime64[ns]
Recency                2240 non-null int64
MntWines               2240 non-null int64
MntFruits              2240 non-null int64
MntMeatProducts        2240 non-null int64
MntFishProducts        2240 non-null int64
MntSweetProducts       2240 non-null int64
MntGoldProds           2240 non-null int64
NumDealsPurchases      2240 non-null int64
NumWebPurchases        2240 non-null int64
NumCatalogPurchases    2240 non-null int64
NumStorePurchases      2240 non-null int64
NumWebVisitsMonth      2240 non-null int64
AcceptedCmp3           2240 non-null int64
AcceptedCmp4           2240 non-null int64
AcceptedCmp5           2240 non-null int64
AcceptedCmp1           2240 non-null int64
AcceptedCmp2           2240 non-null int64
Response               2240 non-null int64
Complain               2240 non-null int64
Country                2240 non-null object
dtypes: datetime64[ns](1), int64(24), object(3)
memory usage: 507.5+ KB
```

Final step of Wrangling: Store data

In [18]:
```
# store the file
df_copy.reset_index(drop=True)
df_copy.to_csv('clean_df.csv', index=False)
```

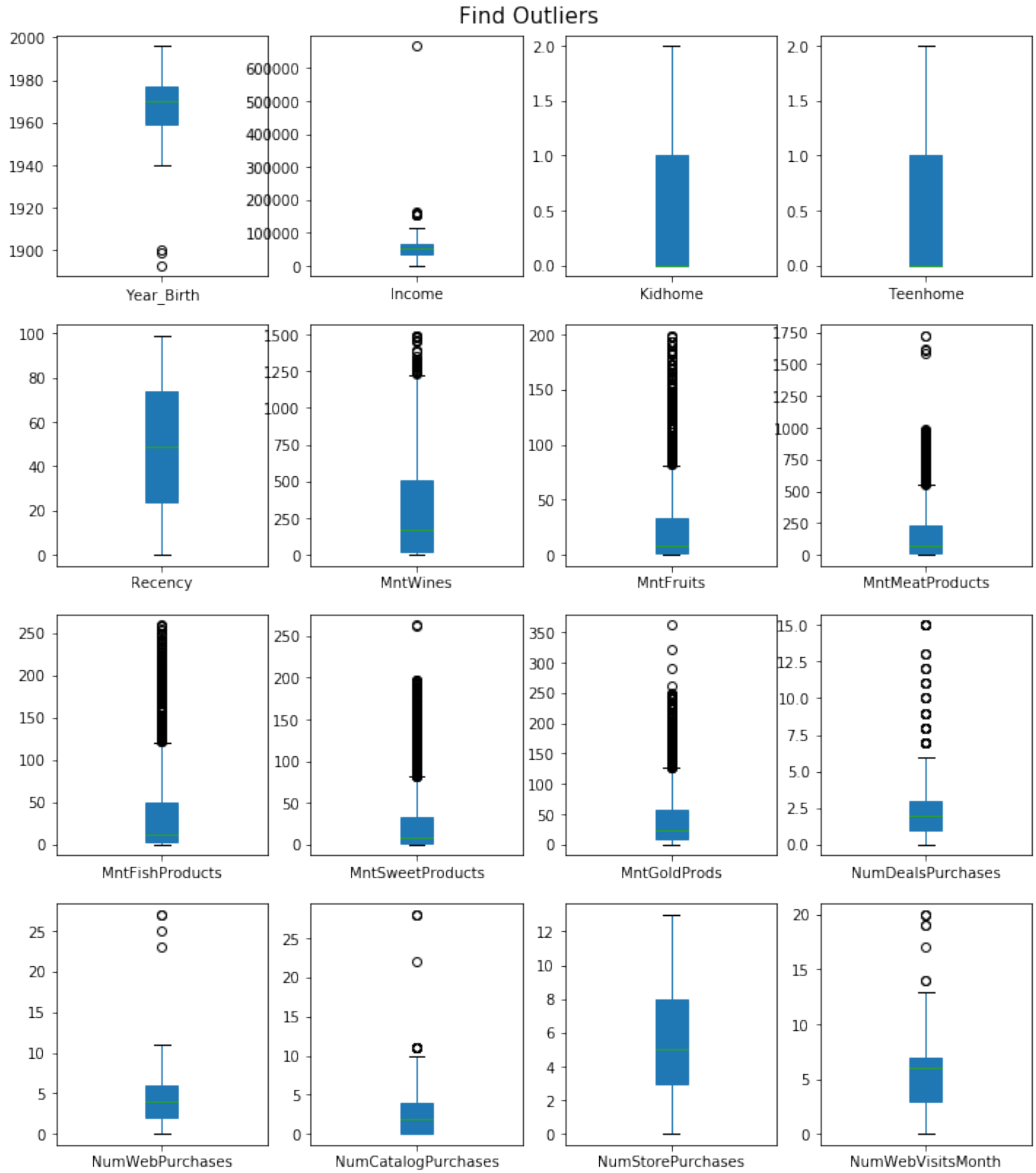
In [55]:
```
#load data
df = pd.read_csv('clean_df.csv')
```

Section 01: Exploratory Data Analysis

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 columns):
ID                     2240 non-null int64
Year_Birth             2240 non-null int64
Education              2240 non-null object
Marital_Status         2240 non-null object
Income                 2240 non-null int64
Kidhome                2240 non-null int64
Teenhome               2240 non-null int64
Dt_Customer            2240 non-null object
Recency                2240 non-null int64
MntWines               2240 non-null int64
MntFruits              2240 non-null int64
MntMeatProducts        2240 non-null int64
MntFishProducts        2240 non-null int64
MntSweetProducts       2240 non-null int64
MntGoldProds           2240 non-null int64
NumDealsPurchases      2240 non-null int64
NumWebPurchases        2240 non-null int64
NumCatalogPurchases    2240 non-null int64
NumStorePurchases      2240 non-null int64
NumWebVisitsMonth      2240 non-null int64
AcceptedCmp3           2240 non-null int64
AcceptedCmp4           2240 non-null int64
AcceptedCmp5           2240 non-null int64
AcceptedCmp1           2240 non-null int64
AcceptedCmp2           2240 non-null int64
Response               2240 non-null int64
Complain               2240 non-null int64
Country                2240 non-null object
dtypes: int64(24), object(4)
memory usage: 490.1+ KB
```

In [56]:
```
# See if there is any outliers

# select columns to plot
df_to_plot = df.drop(columns=['ID', 'AcceptedCmp1', 'AcceptedCmp2', 'Accepted
# subplots
df_to_plot.plot(subplots=True, layout=(4,4), kind='box', figsize=(12,14), pat
plt.suptitle('Find Outliers', fontsize=15, y=0.9)
plt.savefig('boxplots.png', bbox_inches='tight')
```


Find Outliers
Year_Birth
Income
Kidhome
Teenhome
Recency
MntWines
MntFruits
MntMeatProducts
MntFishProducts
MntSweetProducts
MntGoldProds
NumDealsPurchases
NumWebPurchases
NumCatalogPurchases
NumStorePurchases
NumWebVisitsMonth

Section 1-1: Are there any null values or outliers? How will you wrangle/handle them?

- Income has 23 null values, and I used the median number to fill in.
- There are many columns having outliers, but most of them seem like natural outliers came from population, whereas the outliers in Year_birth seems like entry errors since it's impossible that people who was born before 1900 still alive. Therefore, I will remove the outliers in Year_birth. (Reference: https://statisticsbyjim.com/basics/remove-outliers/)

In [57]:

```
df.Year_Birth.describe()
```

Out[57]:

```
count    2240.000000
mean     1968.805804
std        11.984069
min      1893.000000
25%      1959.000000
50%      1970.000000
75%      1977.000000
max      1996.000000
Name: Year_Birth, dtype: float64
```

In [58]:

```
# Remove outliers in year_birth
new_df = df[df.Year_Birth >= (df.Year_Birth.mean()-3*df.Year_Birth.std())]
new_df.Year_Birth.describe()
```

Out[58]:

```
count    2237.000000
mean     1968.901654
std        11.701917
min      1940.000000
25%      1959.000000
50%      1970.000000
75%      1977.000000
max      1996.000000
Name: Year_Birth, dtype: float64
```

Section 1-2: Are there any useful variables that you can engineer with the given data?

- Join_year: The year that person became a customer, which can be engineered from "Dt_Customer"
- Join_month: The month that person became a customer, which can be engineered from "Dt_Customer"
- Join_weekday: The day of the week that person became a customer, which can be engineered from "Dt_Customer"
- Minorhome: The total amount of minors in their family, which can be acquired by summing up by Kidhome and Teenhome.
- Total_Mnt: Total amount spent in the last two years, which can be acquired by summing up all the "Mnt"-related columns
- Total_num_purchase: Total number of purchases in the last two years, which can be acquired by summing up all the "Num"-related columns
- Total_accept: Total amount a customer accepted the offer in marketing campaign, which can be acquired by summing up all the "Accepted"-related columns and the "Response" column
- "AOV": AOV stands for the average order volumn of each customer, which can be engineerd by dividing Total_Mnt by Total_num_purchase

In [59]:

```
new_df.Dt_Customer = pd.to_datetime(new_df.Dt_Customer)
```

In [60]:

```
# Create new features
new_df["Join_year"] = new_df.Dt_Customer.dt.year
new_df["Join_month"] = new_df.Dt_Customer.dt.month
new_df["Join_weekday"] = new_df.Dt_Customer.dt.weekday
new_df["Minorhome"] = new_df.Kidhome + new_df.Teenhome
new_df['Total_Mnt'] = new_df.MntWines+ new_df.MntFruits+ new_df.MntMeatProduc
new_df['Total_num_purchase'] = new_df.NumDealsPurchases+ new_df.NumWebPurchas
new_df['Total_accept'] = new_df.AcceptedCmp1 + new_df.AcceptedCmp2 + new_df.A
new_df['AOV'] = new_df.Total_Mnt/new_df.Total_num_purchase
```

In [61]:

```
new_df.sample(6)
```

Out[61]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
580	7101	1963	PhD	Widow	52278	0	1	2013-01-25
922	9847	1955	2n Cycle	Married	62972	0	1	2012-08-03
585	2535	1978	Master	Married	88097	1	0	2012-08-18
364	7143	1955	2n Cycle	Together	74805	0	1	2013-11-06
1419	1165	1958	PhD	Single	50729	1	1	2013-05-02
1489	9805	1953	Master	Together	56129	0	1	2013-06-20

Section 1-3: Do you notice any patterns or anomalies in the data? Can you plot them?

We can use a heatmap to see the correlations between each variable. When it gets bluer, it means they are positively correlated, and when it gets redder, they are negatively correlated.

Patterns:

1. High Income People

    - tend to spend more and purchase more.
    - tend to visit the company's website less frequently than other people.
    - tend to has few number of purchases made with a discount
2. People having kids at home

    - tend to spend less and purchase less.
    - tend to has high number of purchases made with a discount

1. People who purchased with high average order volumne
    - tend to buy more wines and meat products
    - tend to make high number of purchases made using a catalog
    - tend to not visit the company's website.

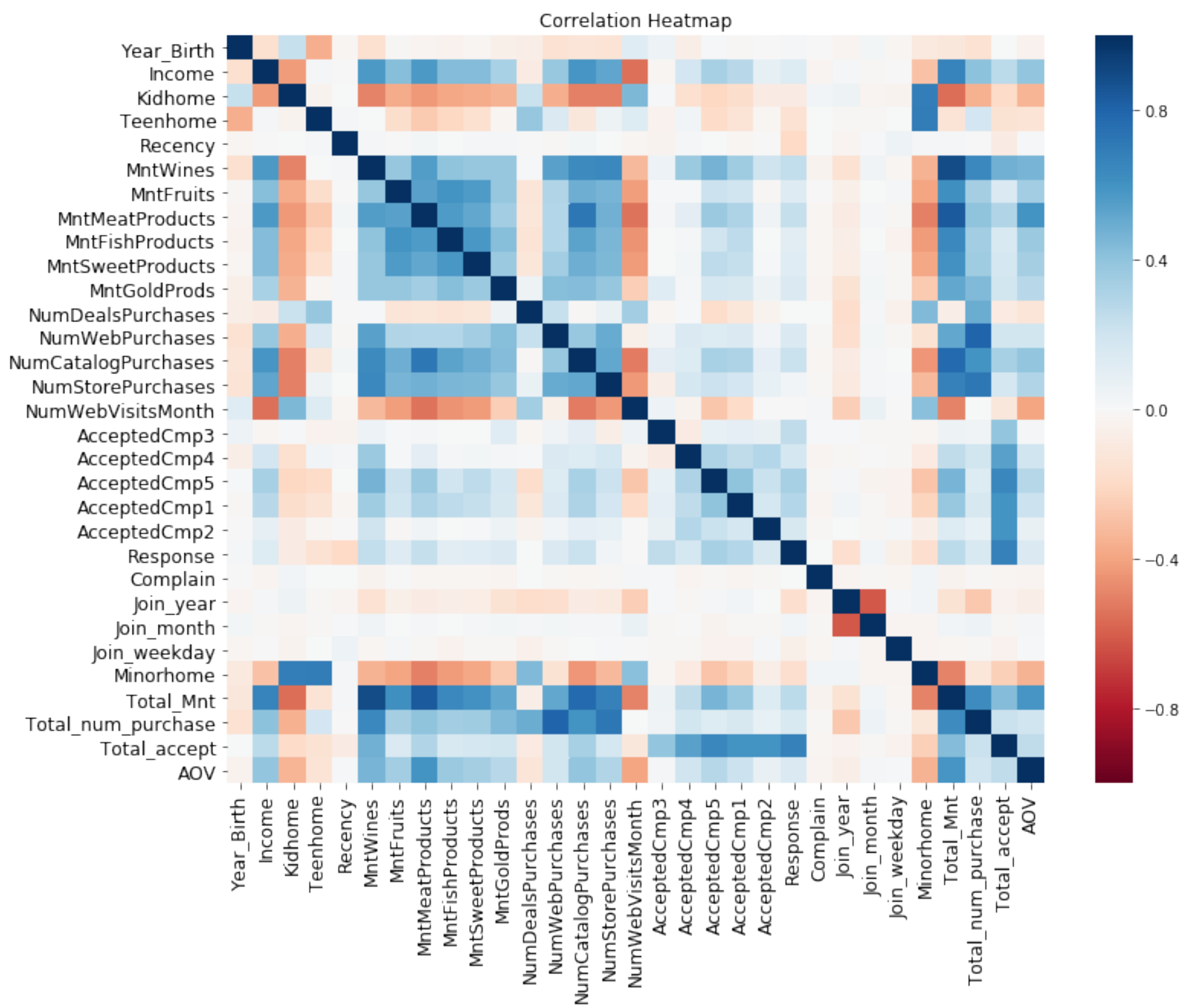
Anomalies:

1. Intuitively, I'd think the more complaints a customer has, the less he/she may spend on our store, but the number of complain in the last two years has almost no correlation with the total amount spent in the last two years => After further investigating the data, I found that it is because we only have 20 customers who complained in the last two years, but we have 2200 customers in total. The customer service in the company has done a wonderful job in the last two years.

See the correlation between variables

In [62]:

```
# select columns to plot
df_to_plot = new_df.drop(columns=['ID'])

# create heatmap
plt.figure(figsize = (12, 9))
s = sb.heatmap(df_to_plot.corr(),  cmap = 'RdBu',vmin = -1, vmax = 1,center =
s.set_yticklabels(s.get_yticklabels(), rotation = 0, fontsize = 12)
s.set_xticklabels(s.get_xticklabels(), rotation = 90, fontsize = 12)
bottom, top = s.get_ylim()
s.set_ylim(bottom + 0.5, top - 0.5)
plt.title("Correlation Heatmap")
plt.savefig('heatmap.png', bbox_inches='tight')
plt.show()
```



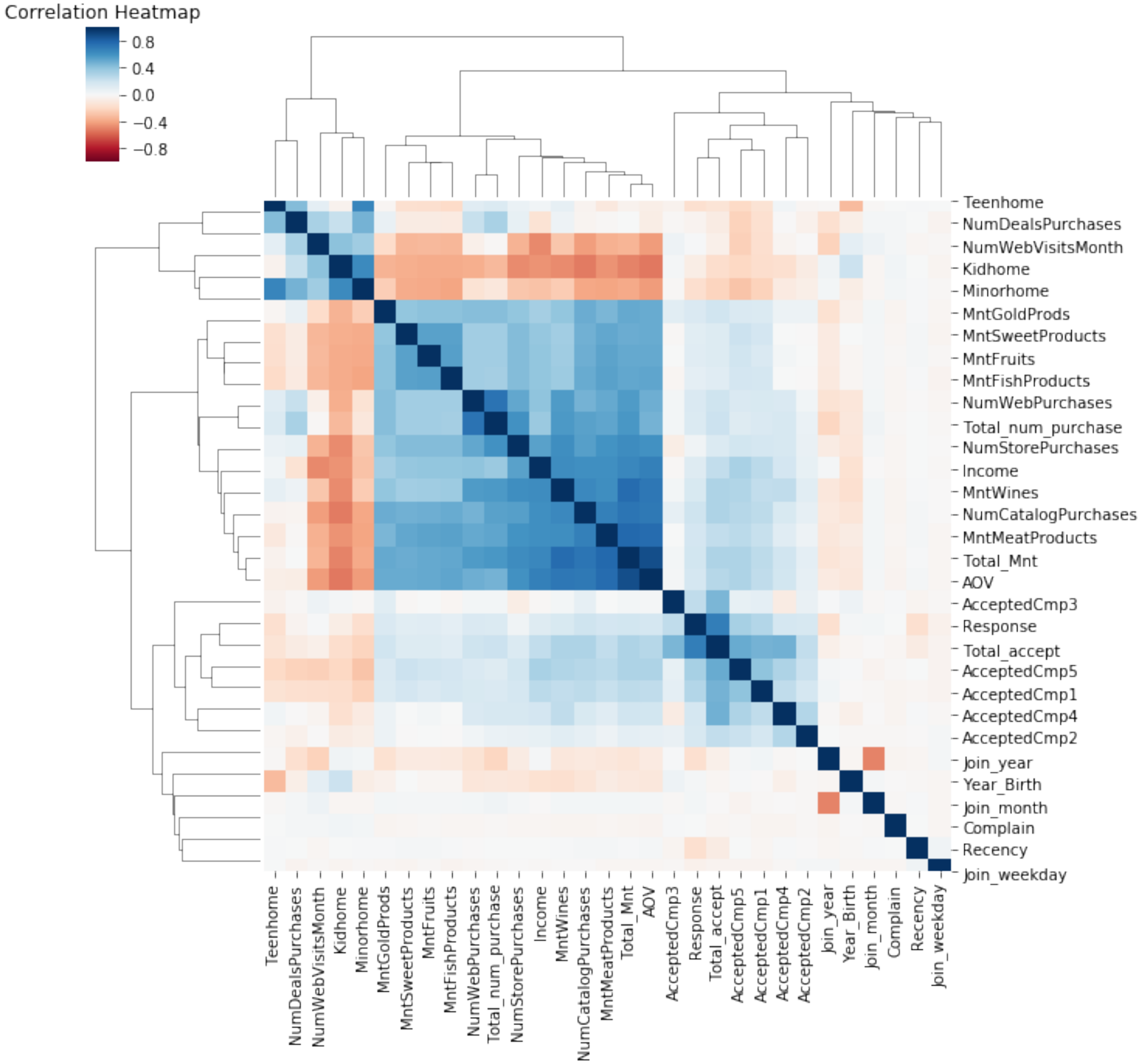
In [63]:

```
new_df.columns
```

Out[63]:
```
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', 'Join_year',
       'Join_month', 'Join_weekday', 'Minorhome', 'Total_Mnt',
       'Total_num_purchase', 'Total_accept', 'AOV'],
      dtype='object')
```

In [64]:
```python
# select columns to plot
df_to_plot = new_df.drop(columns=['ID'])

plt.figure(figsize = (30, 20))
s = sb.clustermap(df_to_plot.corr(method = 'kendall'),  cmap = 'RdBu',vmin = 

plt.title("Correlation Heatmap")
plt.show()
```

```
<Figure size 2160x1440 with 0 Axes>
```



Anomoly

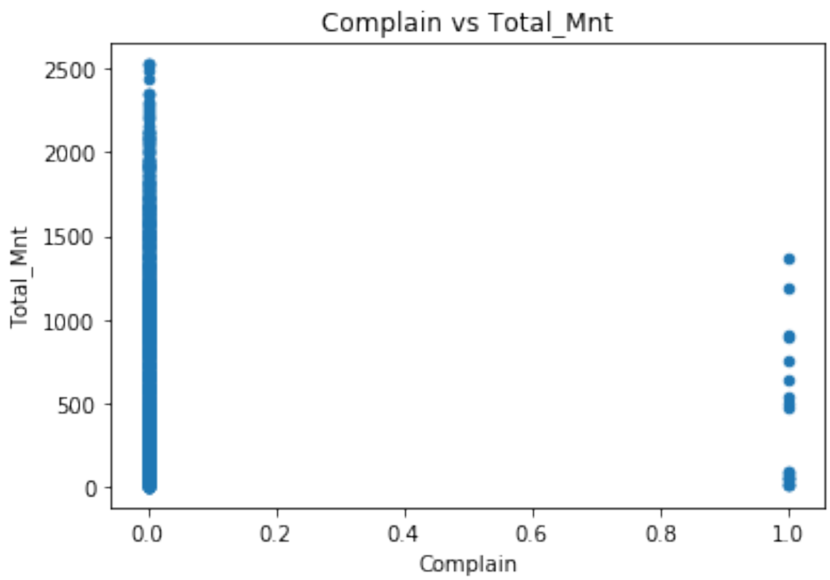
Intuitively, I'd think the more complaints a customer has, the less he/she may spend on our store, but the number of complain in the last two years has almost no correlation with the total amount spent in the last two years

In [65]:

```
new_df.columns
```

Out[65]:
```
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', 'Join_year',
       'Join_month', 'Join_weekday', 'Minorhome', 'Total_Mnt',
       'Total_num_purchase', 'Total_accept', 'AOV'],
      dtype='object')
```

In [66]:
```
# Visualize NumWebPurchases vs NumWebVisitsMonth
new_df.plot(x='Complain', y='Total_Mnt', kind='scatter')
plt.title("Complain vs Total_Mnt");
```



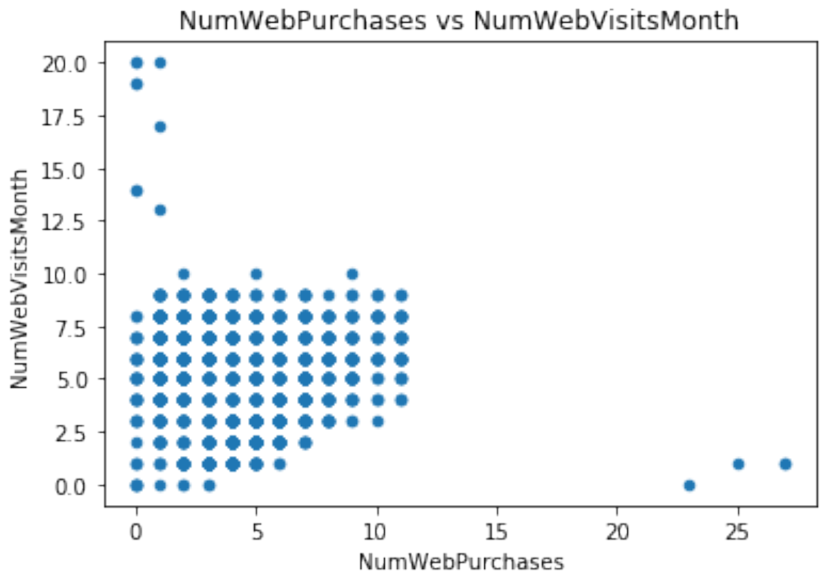
In [67]:
```
from scipy.stats import pearsonr

r, p_value  =  pearsonr(x=new_df['Complain'], y=new_df['Total_Mnt'])

# print results
print('Pearson correlation (r): ', r)
print('Pearson p-value: ', p_value)
```

```
Pearson correlation (r):  -0.03373965091266399
Pearson p-value:  0.11063526070950919
```

In [68]:
```
new_df[new_df.Complain > 0].ID.nunique()
```

Out[68]:
```
20
```

In [69]:
```
# Visualize NumWebPurchases vs NumWebVisitsMonth
new_df.plot(x='NumWebPurchases', y='NumWebVisitsMonth', kind='scatter')
plt.title("NumWebPurchases vs NumWebVisitsMonth");
```



> Indeed, the scatter plot of NumWebPurchases vs NumWebVisitsMonth doesn't show any correlation.

Section 2-1: What factors are significantly related to the number of store purchases?

We can use random forest to predict the number of store purchases and then use the model's feature importance score to rank the factors.

> Top 7 factors are
>
> ```
> 1. Total amount spent in the last two years
> 2. Average order volume
> 3. Total number of purchases in the last two years
> 4. Amount spent on wine in the last 2 years
> 5. Number of purchases made using a catalog
> 6. Number of visits to company's web site in the last month
> 7. Total number of purchases through website in the last two
> years
> ```

In [70]:
```
new_df.NumStorePurchases.hist()
plt.title("Distribution of the number of store purchases");
```

In [71]:

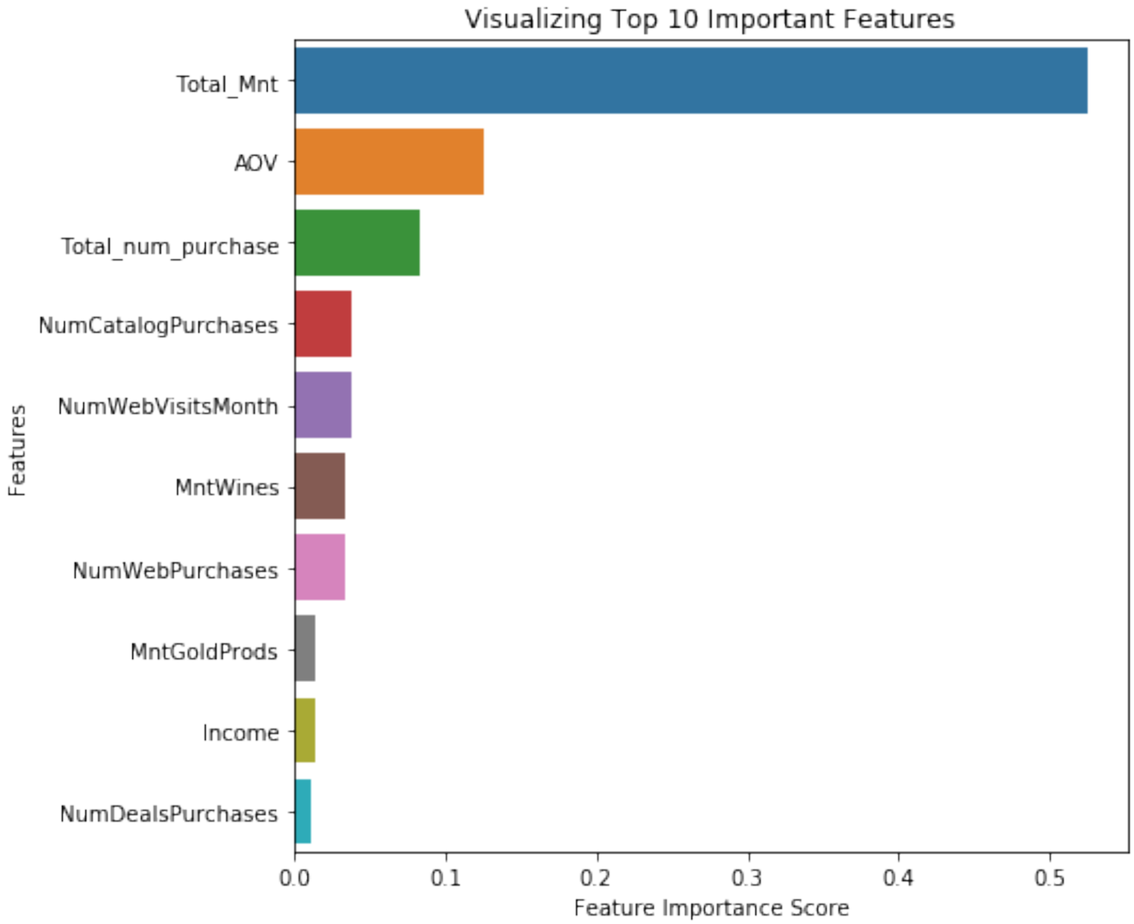
```
# drop ID as everyone has unique ID
rd_df = new_df.drop(columns=['ID', 'Dt_Customer'])
rd_df.replace([np.inf, -np.inf], 0, inplace=True)

# One-hot encoding
rd_df = pd.get_dummies(rd_df)

# Import train_test_split function
from sklearn.model_selection import train_test_split

X=rd_df.drop(columns=['NumStorePurchases'])  # Features
y=rd_df['NumStorePurchases']  # Labels

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
# 70% training and 30% test

#Import Random Forest Model
from sklearn.ensemble import RandomForestRegressor

#Create a Random Forest Classifier with 100 trees
rg = RandomForestRegressor(n_estimators=200, n_jobs=-1)

#Train the model using the training sets y_pred=clf.predict(X_test)
rg.fit(X_train, y_train)

y_pred=rg.predict(X_test)

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
```

```
Mean Absolute Error: 0.7627976190476191
Mean Squared Error: 1.2517511904761904
Root Mean Squared Error: 1.1188168708399915
```

> Finding: The range of NumStorePurchases is 13, and the Root Mean Squared Error is only 1.1(less than 10% of the range), which means it is a relaible model.

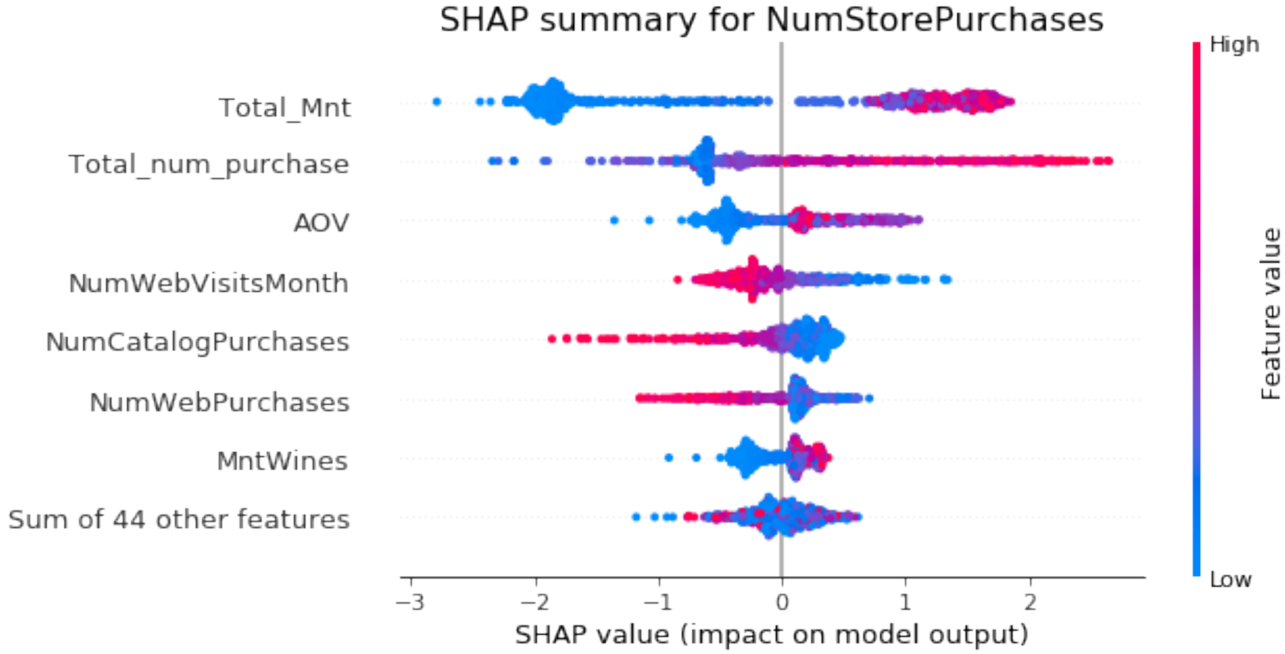
In [72]:

```python
# find feature importance scores
import pandas as pd
feature_imp = pd.Series(rg.feature_importances_,
        index = list(X.columns)).sort_values(ascending=False)

feature_imp = feature_imp[:10]

# Creating a bar plot
plt.figure(figsize = (7, 7))
sb.barplot(x=feature_imp[:10], y=feature_imp.index[:10])
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Top 10 Important Features")
plt.savefig('important_feautres.png', bbox_inches='tight')
plt.show()
```



In [73]:
```
import shap

# calculate shap values
ex = shap.Explainer(rg, X_train)
shap_values = ex(X_test)

# plot
plt.title('SHAP summary for NumStorePurchases', size=16)
fig = shap.plots.beeswarm(shap_values, max_display=8)
plt.savefig('SHAP.png', bbox_inches='tight')
plt.show()
```

```
97%|==================== | 651/672 [00:34<00:01]
```

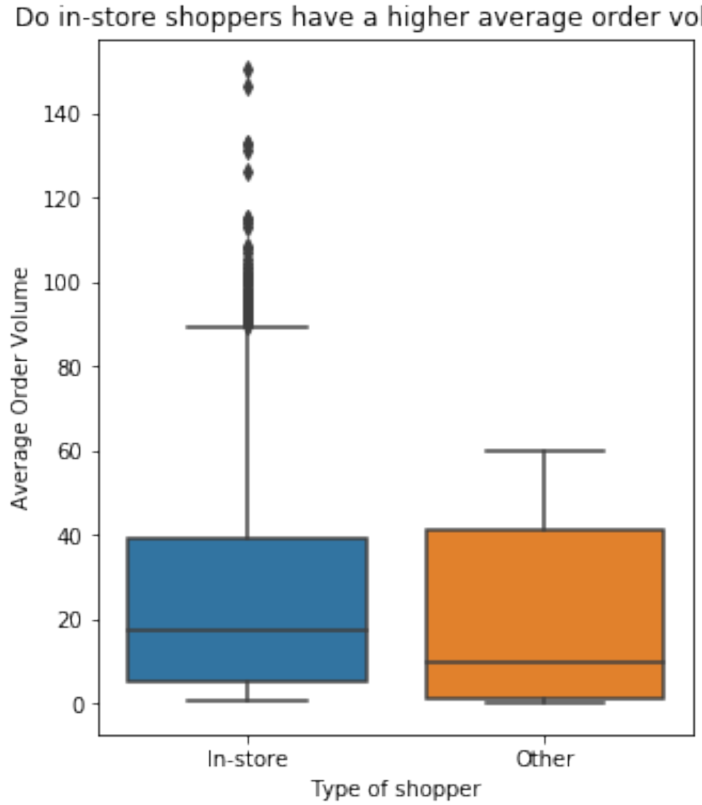



```
<Figure size 432x288 with 0 Axes>
```

> Finding:
>
> `1. The number of store purchase increases with higher total amount spent, higher total purchase amount, higher AOV, and higher amount of wines purchases.`
> `2. The number of store purchase decreases with higher number of website visits, higher number of purchases through catalog, and higher number of purchases through websites.`
>
> Summary: People who mostly shop at store tend to buy more wines, have higher average order volumne, and shop less through internet of catalog.

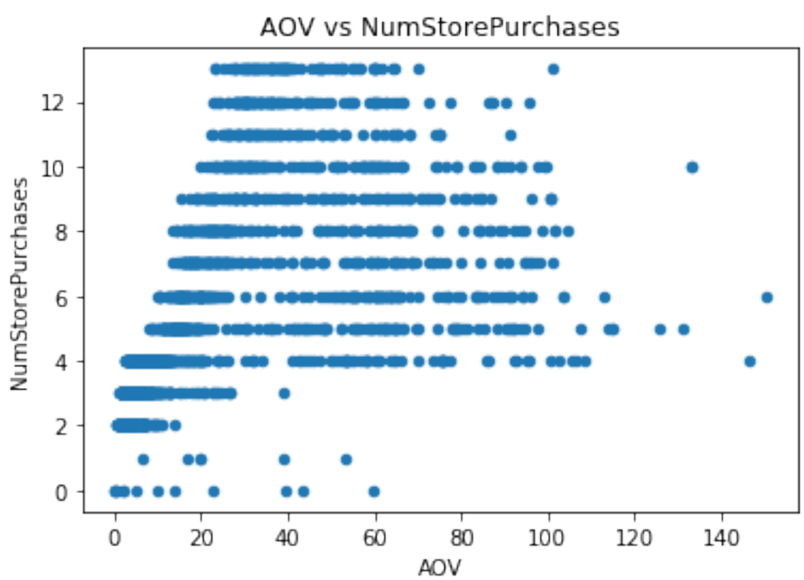
In [75]:

```
store_shoppers = new_df[new_df.NumStorePurchases>0]
store_shoppers = store_shoppers[store_shoppers.AOV <= (store_shoppers.AOV.mea
store_shoppers['Type of shopper'] = "In-store"
other_shoppers = new_df[new_df.NumStorePurchases==0]
other_shoppers['Type of shopper'] = "Other"

plt.figure(figsize = (5, 6))
all_shoppers = store_shoppers.append(other_shoppers)
plt.title("Do in-store shoppers have a higher average order volume?")
sb.boxplot(data = all_shoppers, x = 'Type of shopper', y = 'AOV')
plt.ylabel("Average Order Volume")
plt.savefig('AOV.png', bbox_inches='tight')
```



In [76]:

```
# Visualize MntGoldProds vs NumStorePurchases
all_shoppers.plot(x='AOV', y='NumStorePurchases', kind='scatter')
plt.title("AOV vs NumStorePurchases");
plt.savefig('AOV vs NumStorePurchases.png', bbox_inches='tight')
```



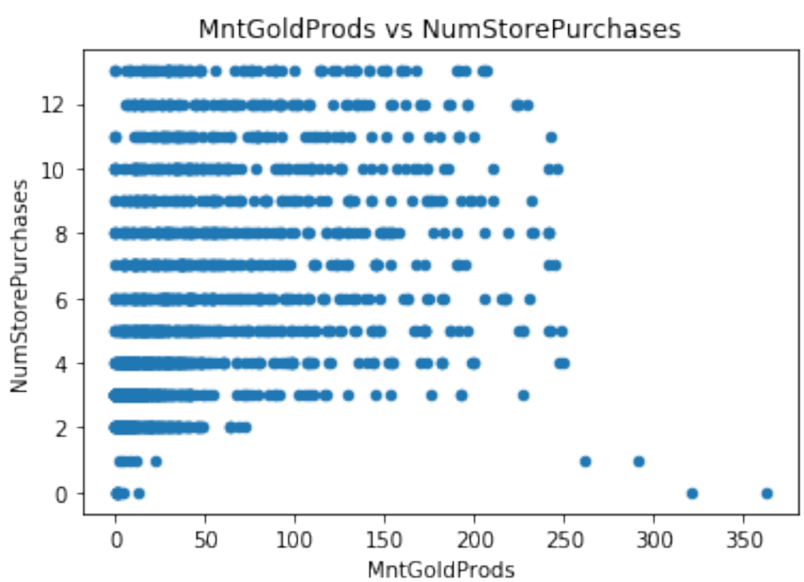
In [77]:
```
from scipy.stats import pearsonr
all_shoppers.replace([np.inf, -np.inf], 0, inplace=True)
r, p_value  =  pearsonr(x=all_shoppers['AOV'], y=all_shoppers['NumStorePurcha

# print results
print('Pearson correlation (r): ', r)
print('Pearson p-value: ', p_value)
```

```
Pearson correlation (r):  0.5505389394031128
Pearson p-value:  2.0526348645442993e-177
```

Section 2-2: Your supervisor insists that people who buy gold are more conservative. Therefore, people who spent an above average amount on gold in the last 2 years would have more in store purchases. Justify or refute this statement using an appropriate statistical test

Yes, they are statistically significant that they have positive correlation

In [78]:
```
# Visualize MntGoldProds vs NumStorePurchases
new_df.plot(x='MntGoldProds', y='NumStorePurchases', kind='scatter')
plt.title("MntGoldProds vs NumStorePurchases");
plt.savefig('MntGoldProds vs NumStorePurchases.png', bbox_inches='tight')
```



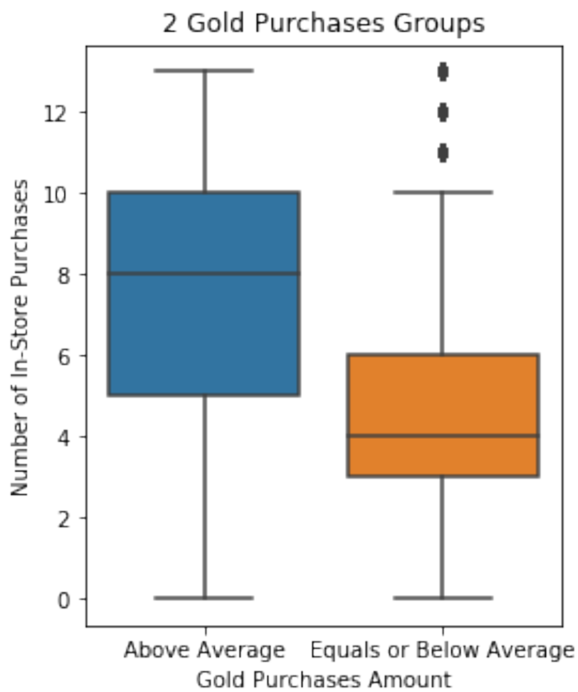
In [79]:

```
from scipy.stats import pearsonr

r, p_value  =  pearsonr(x=new_df['MntGoldProds'], y=new_df['NumStorePurchases

# print results
print('Pearson correlation (r): ', r)
print('Pearson p-value: ', p_value)
```

```
Pearson correlation (r):  0.38326418634704296
Pearson p-value:  3.4668974417790955e-79
```

In [80]:

```
gold_above_avg =  new_df[new_df.MntGoldProds > new_df.MntGoldProds.mean()]
gold_above_avg['Gold Purchases Amount'] = "Above Average"
gold_equ_or_below_avg =  new_df[new_df.MntGoldProds <= new_df.MntGoldProds.me
gold_equ_or_below_avg['Gold Purchases Amount'] = "Equals or Below Average"

plt.figure(figsize = (4, 5))
df_gold = gold_above_avg.append(gold_equ_or_below_avg)
plt.title("2 Gold Purchases Groups")
sb.boxplot(data = df_gold, x = 'Gold Purchases Amount', y = 'NumStorePurchase
plt.ylabel("Number of In-Store Purchases");
```



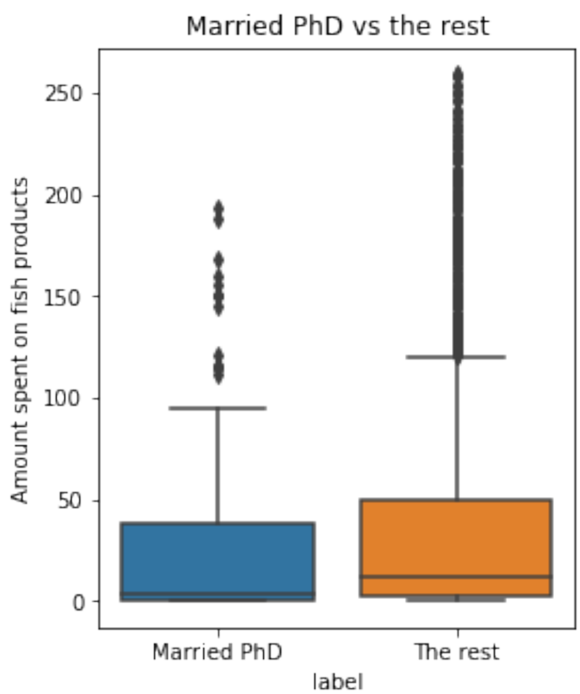
Section 2-3: Fish has Omega 3 fatty acids which are good for the brain. Accordingly, do "Married PhD candidates" have a significant relation with amount spent on fish?

Married PhD spends less on fish products than the rest.

In [81]:
```
new_df.columns
```

Out[81]:
```
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', 'Join_year',
       'Join_month', 'Join_weekday', 'Minorhome', 'Total_Mnt',
       'Total_num_purchase', 'Total_accept', 'AOV'],
      dtype='object')
```

In [82]:
```
# divide the data into two groups: married PhD and the rest
married_phd = new_df[(new_df.Education == "PhD") & (new_df.Marital_Status ==
married_phd['label'] = "Married PhD"
the_rest = new_df[(new_df.Education != "PhD") | (new_df.Marital_Status != "Ma
the_rest['label'] = "The rest"

df_combined = married_phd.append(the_rest)
plt.figure(figsize = (4, 5))
plt.title("Married PhD vs the rest")
sb.boxplot(data = df_combined, x = 'label', y = 'MntFishProducts')
plt.ylabel("Amount spent on fish products");
plt.savefig('Married PhD vs the rest.png', bbox_inches='tight')
```



In [83]:
```
# use t-test to test these two groups have the same mean
from scipy.stats import ttest_ind

#This is a two-sided test for the null hypothesis that 2 independent samples 
#This test assumes that the populations have identical variances by default.
pval = ttest_ind(married_phd.MntFishProducts, the_rest.MntFishProducts).pvalu
print("T-test p-value: ", pval)
```

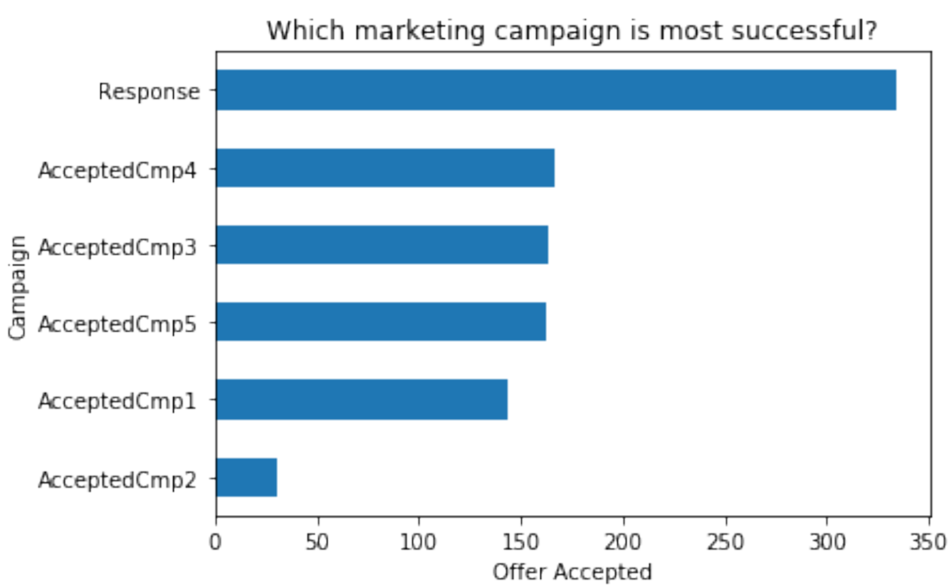
```
T-test p-value:  0.005297012242158541
```

> Note: Since p-value is less than 0.05, I concluded that we reject the null hypothesis, meaning that their means are not only different, but the Married Phd's mean is lower than the rest as we can see from the graph.

Section 3-1: Which marketing campaign is most successful?

The last marketing campaign is most successful.

In [84]:
```
new_df[["AcceptedCmp1", "AcceptedCmp2","AcceptedCmp3","AcceptedCmp4","Accepte
plt.title("Which marketing campaign is most successful?")
plt.xlabel("Offer Accepted");
plt.ylabel("Campaign")
plt.savefig('Which marketing campaign is most successful.png', bbox_inches='t
```



Section 3-2: What does the average customer look like for this company?

An average customer...

- has an annual income of 52200 dollars
- had purchased 49 days ago
- has an AOV of 26.8 dollars
- has spent 605 dollars
- has purchased 20 times
- became a customer in mid-June
- became a customer on Thursday
- spent most on wines(300 dollars) and then meat products(165 dollars)
- spent least on fruit(26 dollars) and sweet products(27 dollars)

In [85]:

```
new_df.replace([np.inf, -np.inf], 0, inplace=True)
```

In [86]:

```
new_df.mean()
```

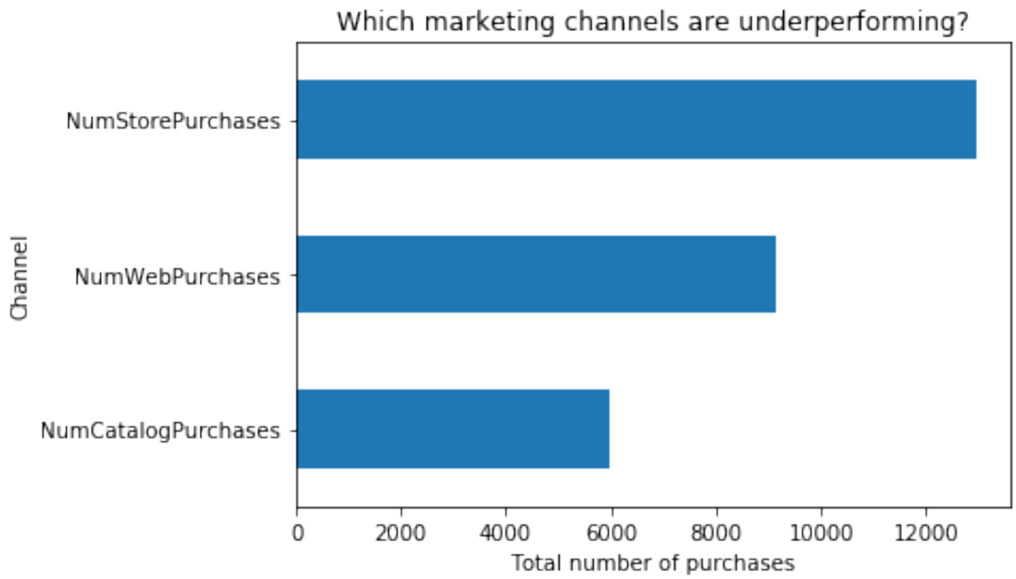
Out[86]:
```
ID                     5590.726419
Year_Birth             1968.901654
Income                52227.402325
Kidhome                   0.444345
Teenhome                  0.506482
Recency                  49.104604
MntWines                303.995530
MntFruits                26.270451
MntMeatProducts         166.916853
MntFishProducts          37.523022
MntSweetProducts         27.068842
MntGoldProds             43.968708
NumDealsPurchases         2.326777
NumWebPurchases           4.087170
NumCatalogPurchases       2.662494
NumStorePurchases         5.794367
NumWebVisitsMonth         5.319177
AcceptedCmp3              0.072865
AcceptedCmp4              0.074654
AcceptedCmp5              0.072418
AcceptedCmp1              0.064372
AcceptedCmp2              0.013411
Response                  0.149307
Complain                  0.008941
Join_year              2013.027716
Join_month                6.465802
Join_weekday              2.988824
Minorhome                 0.950827
Total_Mnt               605.743406
Total_num_purchase       20.189987
Total_accept              0.473849
AOV                      26.842831
dtype: float64
```

Section 3-3: Which marketing channels are underperforming?

Catalog is the most underperforming channel.

In [87]:
```
new_df[["NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases"]].sum()
plt.title("Which marketing channels are underperforming?")
plt.xlabel("Total number of purchases")
plt.ylabel("Channel")
```

Out[87]:
```
Text(0, 0.5, 'Channel')
```



Further Investigation:

Now that we know the last campaign is the most successful one, we can further investigate the differences in the customer characteristics and purchases behaviors(listed below) between the most succesful campaing, the last one, and the rest of the campaigns, the campaign 1-5.

- Characteristics: 'Year_Birth', 'Income', 'Minorhome', 'Country', 'Join_month', 'Join_weekday'
- Purchase behaviors:
  - Products: 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts'
  - Channel: 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases'
  - Total: 'Total_Mnt', 'Total_num_purchase', 'AOV'

In [88]:
```
# create 2 groups that accepted the offers from the last campaign and the cam
cp_last = new_df[new_df.Response > 0]
cp__the_rest = new_df[new_df.AcceptedCmp2 == 0]

cp_last.shape[0], cp__the_rest.shape[0]
```

Out[88]:
```
(334, 2207)
```

In [89]:
```
new_df.Country.value_counts()
```

Out[89]:
```
SP     1094
SA      336
CA      268
AUS     160
IND     147
GER     120
US      109
ME        3
Name: Country, dtype: int64
```

In [90]:
```
# remove the overlapping customers who accepted offers from both cp_last and
# so that twe can see the clear differences between these two groups
cp__the_rest2 = cp__the_rest
for i in list(cp__the_rest.ID):
    if i in list(cp_last.ID):
        cp__the_rest2 = cp__the_rest2[cp__the_rest2.ID != i]

cp_last.shape[0], cp__the_rest2.shape[0]
```

Out[90]:
```
(334, 1893)
```

In [91]:
```
cp_last = cp_last[['Year_Birth', 'Income', 'Minorhome', 'Country', 'Join_mont
                  'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProduct
                  'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchase
                  'Total_Mnt', 'Total_num_purchase', 'AOV']]
cp__the_rest2 = cp__the_rest2[['Year_Birth', 'Income', 'Minorhome', 'Country'
                  'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProduct
                  'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchase
                  'Total_Mnt', 'Total_num_purchase', 'AOV']]
```

In [92]:
```
cp_last.mean()
```

Out[92]:
```
Year_Birth              1969.416168
Income                 60183.242515
Minorhome                  0.646707
Join_month                 6.739521
Join_weekday               2.700599
MntWines                 502.703593
MntFruits                 38.203593
MntMeatProducts          294.353293
MntFishProducts           52.050898
MntSweetProducts          38.634731
NumWebPurchases            5.071856
NumCatalogPurchases        4.203593
NumStorePurchases          6.095808
Total_Mnt                987.392216
Total_num_purchase        23.000000
AOV                       41.829197
dtype: float64
```

In [93]:

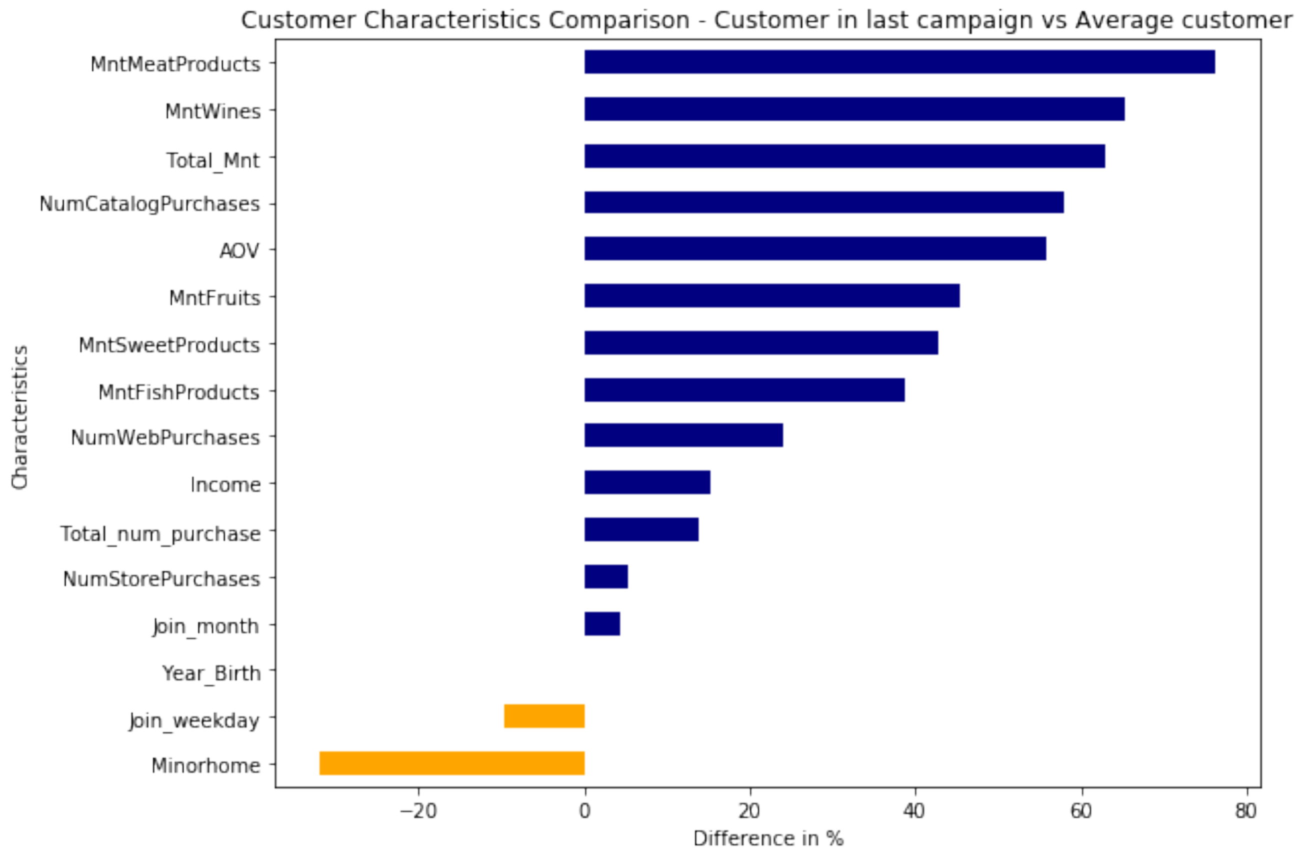
```
new_df2 = new_df[['Year_Birth', 'Income', 'Minorhome', 'Country', 'Join_month
                  'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProduct
                  'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchase
                  'Total_Mnt', 'Total_num_purchase', 'AOV']]
new_df2.mean()
```

Out[93]:

```
Year_Birth              1968.901654
Income                 52227.402325
Minorhome                  0.950827
Join_month                 6.465802
Join_weekday               2.988824
MntWines                 303.995530
MntFruits                 26.270451
MntMeatProducts          166.916853
MntFishProducts           37.523022
MntSweetProducts          27.068842
NumWebPurchases            4.087170
NumCatalogPurchases        2.662494
NumStorePurchases          5.794367
Total_Mnt                605.743406
Total_num_purchase        20.189987
AOV                       26.842831
dtype: float64
```

In [94]:

```
# visualize the differences
plt.figure(figsize = (9, 7))
value1 = pd.DataFrame((((cp_last.mean()) - new_df2.mean()) / new_df2.mean())*
value1.dropna(inplace = True)
value1.sort_values(by=0,inplace = True)
value1['positive'] = value1[0] >=0
value1[0].plot(kind='barh', color=value1.positive.map({True: 'navy', False: '
plt.title("Customer Characteristics Comparison - Customer in last campaign vs
plt.xlabel("Difference in %")
plt.ylabel("Characteristics");
```



In [95]:

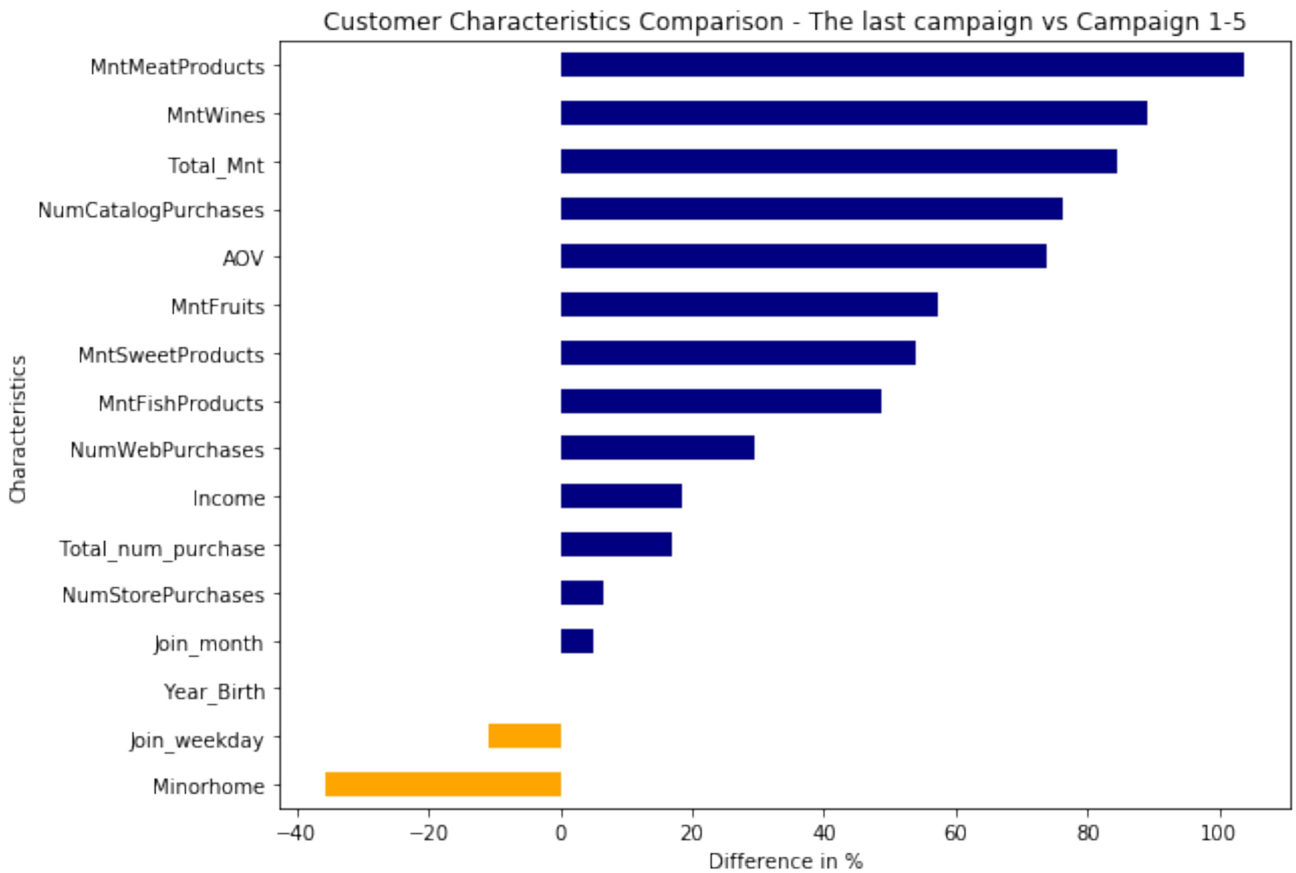
```
cp__the_rest2.mean()
```

Out[95]:

```
Year_Birth                1968.807184
Income                   50753.805600
Minorhome                    1.005283
Join_month                   6.409403
Join_weekday                 3.032752
MntWines                   265.836767
MntFruits                   24.267829
MntMeatProducts            144.358690
MntFishProducts             34.996302
MntSweetProducts            25.112520
NumWebPurchases              3.918119
NumCatalogPurchases          2.384046
NumStorePurchases            5.728473
Total_Mnt                  535.491812
Total_num_purchase          19.681986
AOV                         24.083059
dtype: float64
```

In [96]:

```
# visualize the differences
plt.figure(figsize = (9, 7))
value = pd.DataFrame((((cp_last.mean()) - cp__the_rest2.mean()) / cp__the_res
value.dropna(inplace = True)
value.sort_values(by=0,inplace = True)
value['positive'] = value[0] >=0
value[0].plot(kind='barh', color=value.positive.map({True: 'navy', False: 'or
plt.title("Customer Characteristics Comparison - The last campaign vs Campaig
plt.xlabel("Difference in %")
plt.ylabel("Characteristics")
plt.savefig('Customer Characteristics Comparison - The last campaign vs Campa
```



In [97]:

```
cp_last_country = pd.DataFrame((cp_last.Country.value_counts()/cp_last.shape[
cp_last_country.rename(columns={'Country':'Percent'}, inplace=True)
cp_last_country['country'] = cp_last_country.index
cp_last_country = cp_last_country.sort_values('country')
cp_last_country.drop(['country'], axis=1, inplace=True)
cp_last_country
```

Out[97]:

	Percent
AUS	6.886228
CA	11.377246
GER	5.089820
IND	3.892216
ME	0.598802
SA	15.568862
SP	52.694611
US	3.892216

In [98]:

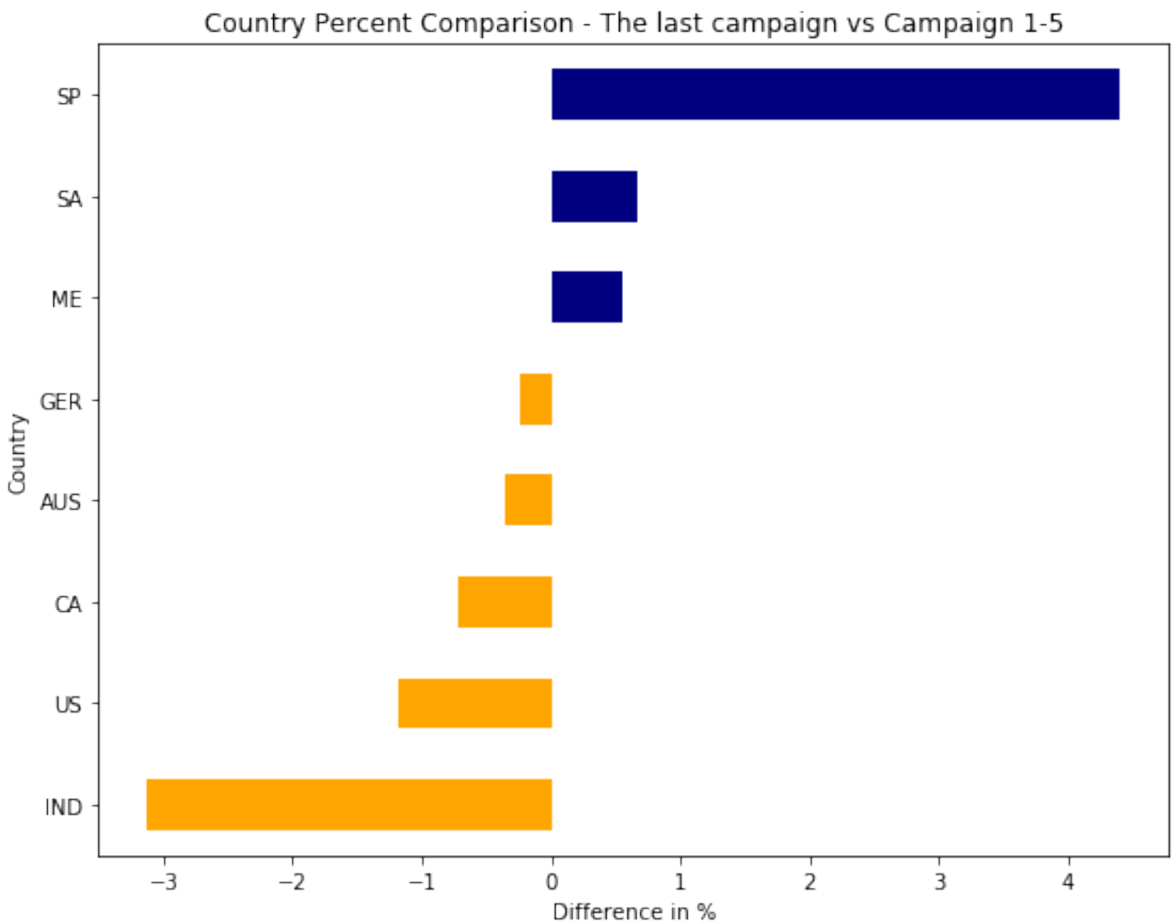
```
cp__the_rest2_country = pd.DataFrame((cp__the_rest2.Country.value_counts()/cp_
cp__the_rest2_country.rename(columns={'Country':'Percent'}, inplace=True)
cp__the_rest2_country['country'] = cp__the_rest2_country.index
cp__the_rest2_country = cp__the_rest2_country.sort_values('country')
cp__the_rest2_country.drop(['country'], axis=1, inplace=True)
cp__the_rest2_country
```

Out[98]:

	Percent
AUS	7.237190
CA	12.097200
GER	5.335446
IND	7.025885
ME	0.052826
SA	14.896989
SP	48.283148
US	5.071315

In [99]:

```
country_final = cp_last_country-cp__the_rest2_country

# visualize the differences
plt.figure(figsize = (9, 7))
country_final.sort_values(by="Percent",inplace = True)
country_final['positive'] = country_final["Percent"] >=0
country_final["Percent"].plot(kind='barh', color=country_final.positive.map({'
plt.title("Country Percent Comparison - The last campaign vs Campaign 1-5")
plt.xlabel("Difference in %")
plt.ylabel("Country")
plt.savefig('Country Percent Comparison - The last campaign vs Campaign 1-5',
```



In [100…

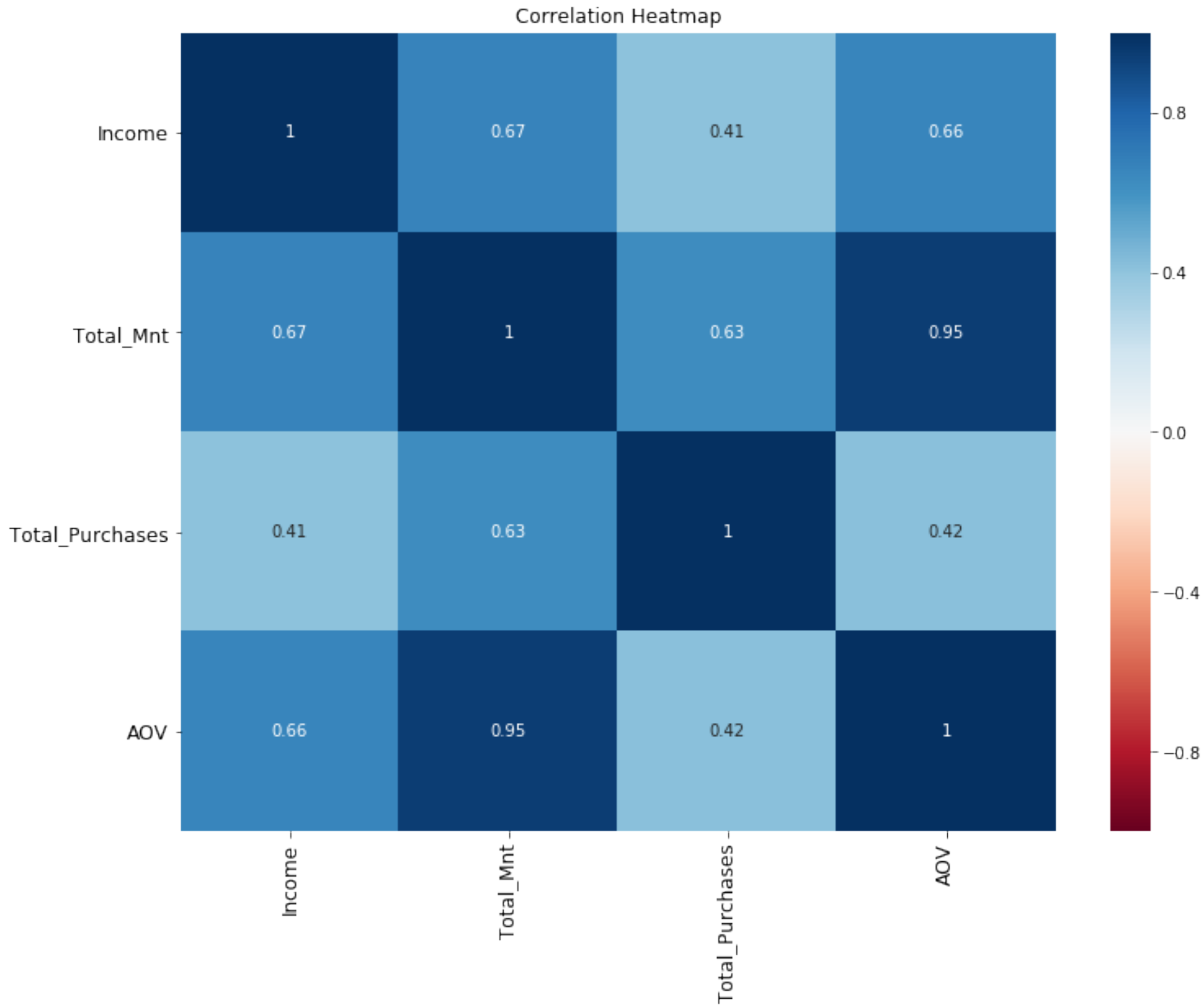
```
new_df.Country.value_counts()
```

Out[100…

```
SP      1094
SA       336
CA       268
AUS      160
IND      147
GER      120
US       109
ME         3
Name: Country, dtype: int64
```

In [101…

```
# select columns to plot
new_df2 = new_df[new_df.AOV <= (new_df.AOV.mean()+3*new_df.AOV.std())]
new_df2.replace([np.inf, -np.inf], 0, inplace=True)
new_df2 = new_df2[new_df2.Total_num_purchase <= (new_df2.Total_num_purchase.m
new_df2 = new_df2[new_df2.Total_Mnt <= (new_df2.Total_Mnt.mean()+3*new_df2.To

df_to_plot = new_df2[['Income', 'Total_Mnt','Total_num_purchase', 'AOV']]
df_to_plot.rename(columns={'Total_num_purchase':'Total_Purchases'}, inplace=T

# create heatmap
plt.figure(figsize = (12, 9))
s = sb.heatmap(df_to_plot.corr(),  annot = True,cmap = 'RdBu',vmin = -1, vmax
s.set_yticklabels(s.get_yticklabels(), rotation = 0, fontsize = 12)
s.set_xticklabels(s.get_xticklabels(), rotation = 90, fontsize = 12)
bottom, top = s.get_ylim()
s.set_ylim(bottom + 0.5, top - 0.5)
plt.title("Correlation Heatmap")
plt.savefig('heatmap2.png', bbox_inches='tight')
plt.show()
```



In [109…

```
new_df.columns
```

Out[109…

```
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', 'Join_year',
       'Join_month', 'Join_weekday', 'Minorhome', 'Total_Mnt',
       'Total_num_purchase', 'Total_accept', 'AOV'],
      dtype='object')
```

In [140…

```
# drop ID as everyone has unique ID
rd_df = new_df.drop(columns=['ID', 'Dt_Customer','AcceptedCmp3', 'AcceptedCmp
       'AcceptedCmp2', 'Response'])
rd_df.replace([np.inf, -np.inf], 0, inplace=True)

# One-hot encoding
rd_df = pd.get_dummies(rd_df)

# Import train_test_split function
from sklearn.model_selection import train_test_split

X=rd_df.drop(columns=['Total_accept'])  # Features
y=rd_df['Total_accept']  # Labels

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
# 70% training and 30% test

#Import Random Forest Model
from sklearn.ensemble import RandomForestRegressor

#Create a Random Forest Classifier with 100 trees
rg2 = RandomForestRegressor(n_estimators=200, n_jobs=-1)

#Train the model using the training sets y_pred=clf.predict(X_test)
rg2.fit(X_train, y_train)

y_pred=rg2.predict(X_test)

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, 
```

```
Mean Absolute Error: 0.5034036458333334
Mean Squared Error: 0.6659027569180823
Root Mean Squared Error: 0.8160286495694146
```

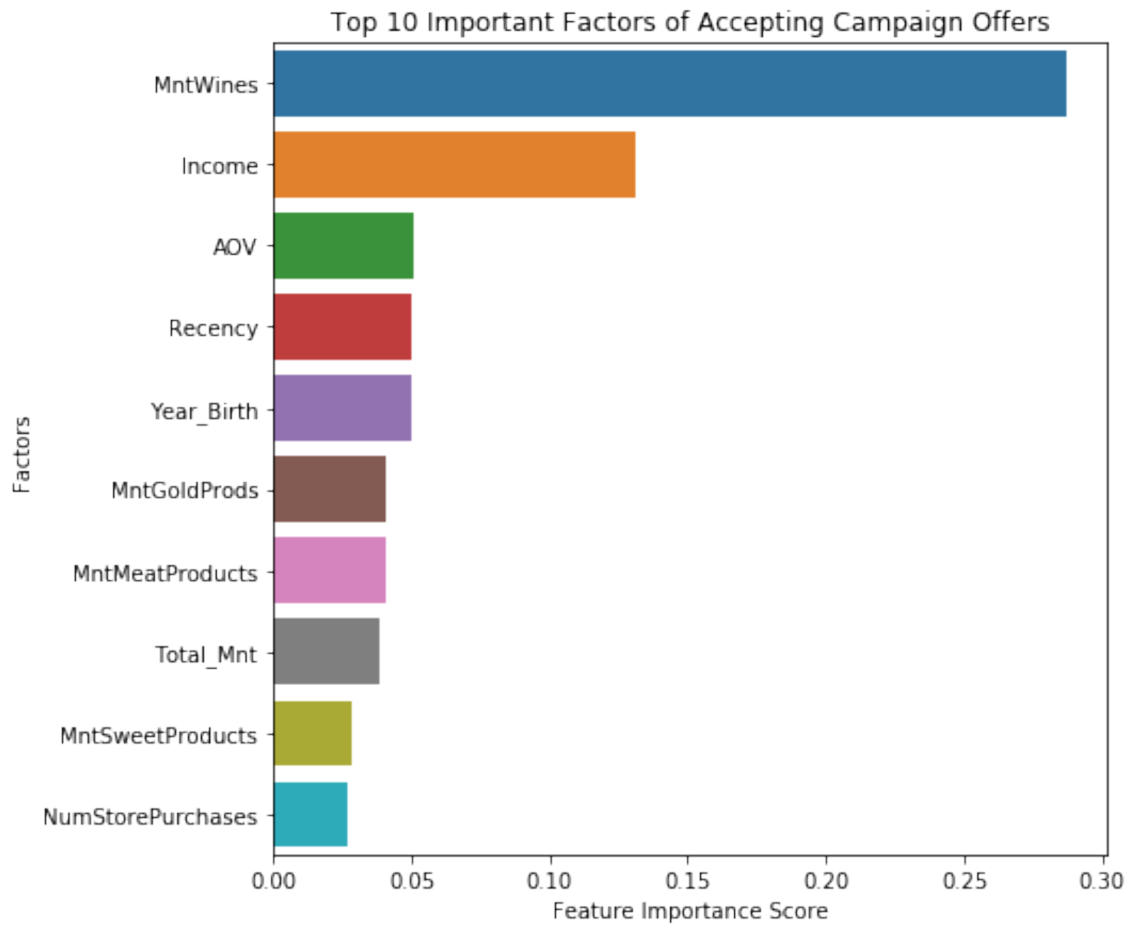
In [139…

```
from scipy.stats import pearsonr

list_ = ['MntWines', 'MntMeatProducts','MntGoldProds', 'MntFishProducts', 'Mn
for i in list_:
    r, p_value  =  pearsonr(x=new_df[i], y=new_df['Total_accept'])
    print(i, "vs Total_accept:")
    # print results
    print('Pearson correlation (r): ', r)
    print('Pearson p-value: ', p_value)
    print(" ")
```

```
MntWines vs Total_accept:
Pearson correlation (r):  0.4787761368904706
Pearson p-value:  1.538453690271273e-128

MntMeatProducts vs Total_accept:
Pearson correlation (r):  0.30103342502969976
Pearson p-value:  4.4065093290159594e-48

MntGoldProds vs Total_accept:
Pearson correlation (r):  0.19068006658523506
Pearson p-value:  9.275051767778576e-20

MntFishProducts vs Total_accept:
Pearson correlation (r):  0.1591461828125727
Pearson p-value:  3.697344094817059e-14

MntFruits vs Total_accept:
Pearson correlation (r):  0.14976326660144543
Pearson p-value:  1.082631664111806e-12

MntSweetProducts vs Total_accept:
Pearson correlation (r):  0.17808985183102097
Pearson p-value:  2.1328375413163002e-17
```

In [142…

```
# find feature importance scores
import pandas as pd
feature_imp = pd.Series(rg2.feature_importances_,
        index = list(X.columns)).sort_values(ascending=False)

feature_imp = feature_imp[:10]

# Creating a bar plot
plt.figure(figsize = (7, 7))
sb.barplot(x=feature_imp[:10], y=feature_imp.index[:10])
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Factors')
plt.title("Top 10 Important Factors of Accepting Campaign Offers")
plt.savefig('important_feautres.png', bbox_inches='tight')
plt.show()
```



In [143…

```
new_df[['Join_month', 'Join_weekday']].mean()
```

Out[143…

```
Join_month      6.465802
Join_weekday    2.988824
dtype: float64
```

Section 04: CMO Recommendations

The goal of this project

I'm a marketing data analyst and I've been told by the Chief Marketing Officer that recent marketing campaigns have not been as effective as they were expected to be. I need to analyze the data set to understand this problem and propose data-driven solutions.

Summaries

- The last campaign performed nearly twice as good as the previous campaigns
  - The last campaign attracted more valuable customers in terms of AOV, the total amount spent, and the total number of purchases, compared to the customers who

were attracted by the previous campaigns.
  - Spain has relatively more customers (+4%) that were attracted to the last campaign, and India has fewer customers (-3%) that were attracted to the previous campaigns
  - In terms of product categories, the customers in the last campaign spent nearly two times more money on meat products and wines compared to the customers in the previous campaigns.
  - In terms of purchasing channels, the customers in the last campaign purchased more evenly through stores, websites, and catalogs, whereas the customers in the previous campaigns mostly purchased through stores and websites.
  - The customers in the last campaign earned 20% more salary than the customers in the previous campaigns.

- Most customers purchase through physical stores, where people tend to spend more amount per purchase. The reason might be the customers had more impulsive purchases when they saw other similar products in stores.

- People having kids at home are less valuable customers as they...
  - tend to purchase less
  - tend to has a high number of purchases made with a discount

- The average customer...
  - became a customer on Thursdays
  - became a customer in Mid-June

Actionable Data-Driven Solutions

On Acquisition:

1. Keep using the same marketing techniques in the last campaign, and with a focus on promoting meat products and wines

2. Try to spend more marketing budget in Spain, and less in India

3. Try to have a brand discount day on Thursday or a brand discount month in June to attract new customers

On Increasing revenue:

1. Have marketing campaigns to convert customers who shop mostly on a website or catalog to in-store purchasers as most in-store purchases have high average order volume.

2. Build a loyalty program to make high-income customers loyal as long as possible