Exploratory Data Analysis of Customer Segmentation and marketing campaigns

Problem Statement

While going through the analysis, it is vital to have in mind an objective. We will attempt to answer the following questions through Visualization to see if we can find contributing factors to the success of the past campaigns.

- Which products are performing best?
- Which channels are underperforming?
- What does the average customer look like for the Company?
- Which marketing campaign is most successful?
- Which Regions perform best?
- Which costumer segment purchase more?
- What does the costumer segment which accepted the last marketing campaign look like?

Importing Libraries

```
#Importing the Libraries
import numpy as np
import pandas as pd
import datetime
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
import matplotlib.pyplot as plt, numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import ListedColormap
```

Reading csv file

```
dset = pd.read_csv("C:/Users/Lenovo/Desktop/GitHub Projects/njimonda.github.io/Exploratory-Data-Analysis-of-marketing-campaigns/marketin
```

```
print("Number of datapoints:", len(dset))
```

```
Number of datapoints: 2240
```

```
dset.head(3)
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurchase
0	1826	1970	Graduation	Divorced	$84,835.00	0	0	6/16/14	0	189	...	6
1	1	1961	Graduation	Single	$57,091.00	0	0	6/15/14	0	464	...	7
2	10476	1958	Graduation	Married	$67,267.00	0	1	5/13/14	0	134	...	5

3 rows × 28 columns

```
dset.tail(3)
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurcha
2237	22	1976	Graduation	Divorced	$46,310.00	1	0	12/3/12	99	185	...	
2238	528	1978	Graduation	Married	$65,819.00	0	0	11/29/12	99	267	...	
2239	4070	1969	PhD	Married	$94,871.00	0	2	9/1/12	99	169	...	

3 rows × 28 columns

```
dset.shape
```

```
(2240, 28)
```

```
dset.describe()
```

	ID	Year_Birth	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	Mn
count	2240.000000	2240.000000	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	166.950000	37.525446	
std	3246.662198	11.984069	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	
min	0.000000	1893.000000	0.000000	0.000000	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	16.000000	3.000000	
50%	5458.500000	1970.000000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	
75%	8427.750000	1977.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	
max	11191.000000	1996.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	

8 rows × 23 columns

```
dset.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
```

```
dset.isnull().sum().sort_values(ascending=False)
```

```
 Income                24
Country                 0
Complain                0
Year_Birth              0
Education               0
Marital_Status          0
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
ID                     0
dtype: int64
```

Clean whitespace and converting datatype

```
# clean up column names that contain whitespace
dset.columns = dset.columns.str.replace(' ', '')
```

```
# transform Income column to a numerical
dset['Income'] = dset['Income'].str.replace('$', '')
dset['Income'] = dset['Income'].str.replace(',', '').astype('float')
```

```
dset["Dt_Customer"] = pd.to_datetime(dset["Dt_Customer"])
#As it's a date, it's better to change to format of datetime
dates = []
for i in dset["Dt_Customer"]:
    i = i.date()
    dates.append(i)
#Dates of the newest and oldest recorded customer
print("The newest customer's enrolment date in the records:",max(dates))
print("The oldest customer's enrolment date in the records:",min(dates))
```

```
The newest customer's enrolment date in the records: 2014-06-29
The oldest customer's enrolment date in the records: 2012-07-30
```

Handling null values or outliers

```
dset['Income'].fillna(dset['Income'].median())
```

```
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
```

```
#To remove the NA values
dset = dset.dropna()
print("The total number of data-points after removing the rows with missing values are:", len(dset))
```

```
The total number of data-points after removing the rows with missing values are: 2216
```

```
dset[dset.isnull()["Income"] == True].count()["ID"]
#There are 24 objects, it's better to get rid of them not to corrupt the whole picture and insights of the data
```

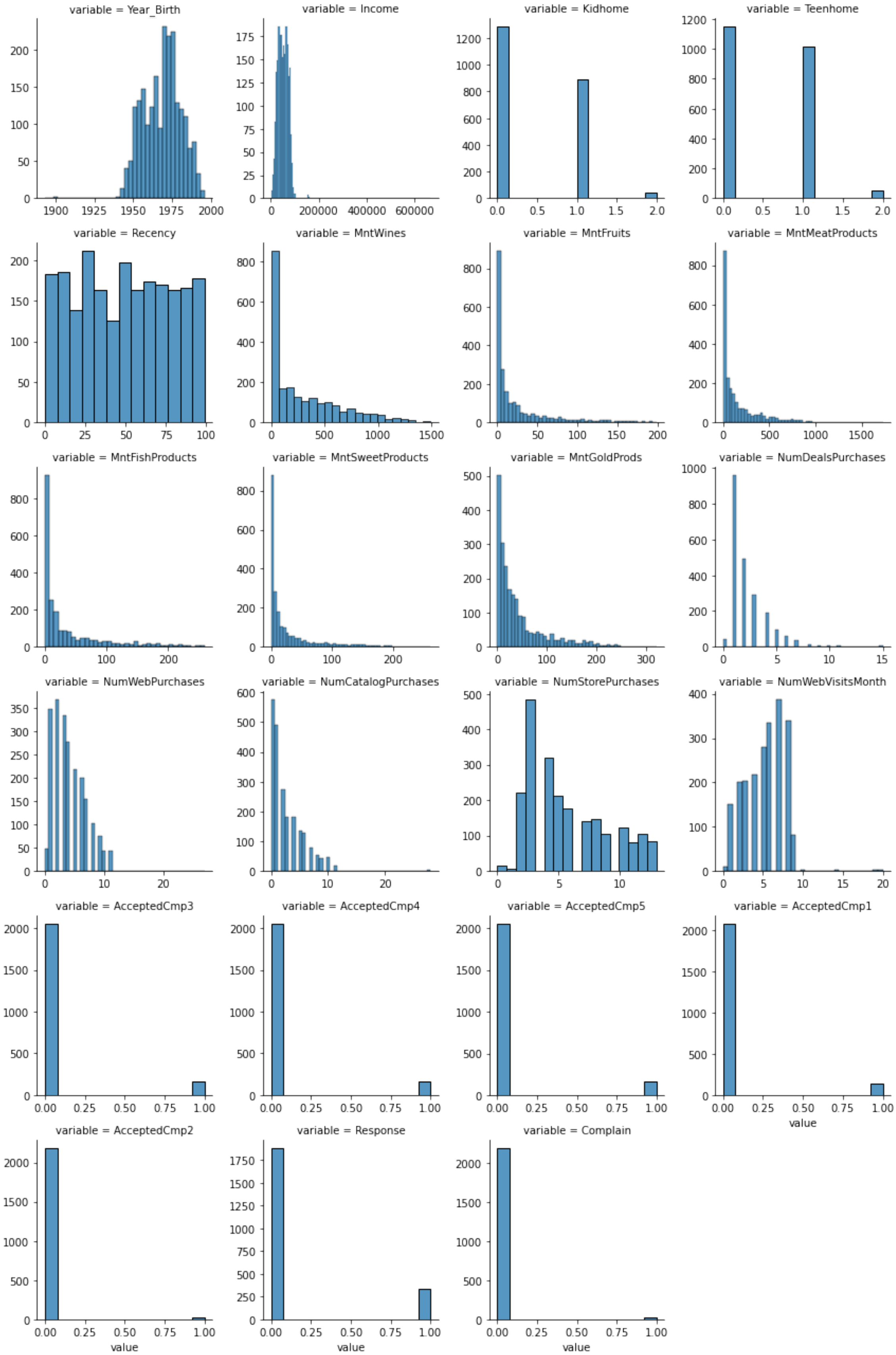
```
0
```

Outliers

```
dset.columns.tolist
```

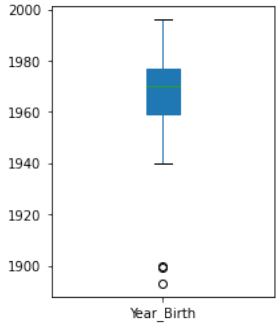
```
<bound method IndexOpsMixin.tolist of Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',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

```
#Show all distributions of all the features
#in the dataset except Country, Education and Martial_Status becouse they are
#objects

dist = pd.DataFrame(data = dset, columns = ['Year_Birth', 'Income', 'Kidhome',
       'Teenhome', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Response', 'Complain'])
nd = pd.melt(dist, value_vars = dist)
n1 = sns.FacetGrid(nd, col = "variable", col_wrap = 4, sharex = False, sharey = False)
n1 = n1.map(sns.histplot, "value")
plt.show()
```



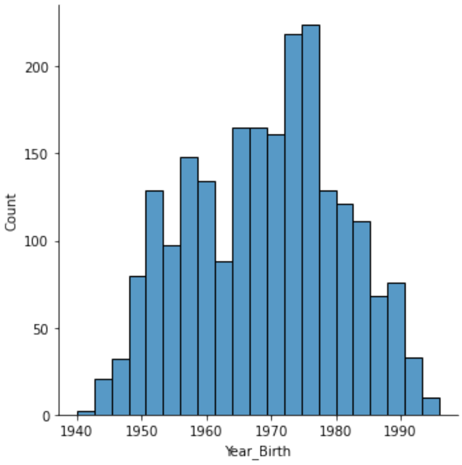
```
dset['Year_Birth'].plot(kind='box', figsize=(3,4), patch_artist=True)
```

```
<AxesSubplot: >
```



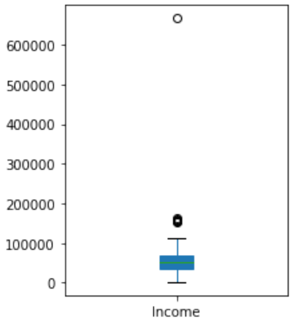
```
dset.drop(dset[dset['Year_Birth'] <= 1900].index, inplace = True) #Drop customers who were born before 1900.
```

```
sns.displot(dset['Year_Birth']) #Distribution of Year_Birth feature
plt.show()
```



```
dset['Income'].plot(kind='box', figsize=(3,4), patch_artist=True)
```

```
<AxesSubplot: >
```

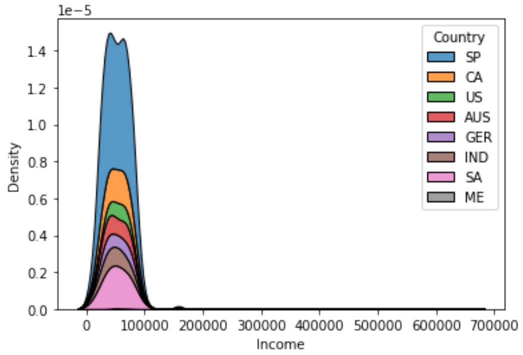


```
dset["Income"].describe() #Quantiles, mean, std of Income feature
```

```
count      2213.000000
mean      52236.581563
std       25178.603047
min        1730.000000
25%       35246.000000
50%       51373.000000
75%       68487.000000
max      666666.000000
Name: Income, dtype: float64
```

```
sns.kdeplot(data=dset, x="Income", hue="Country", multiple="stack")
```

```
<AxesSubplot: xlabel='Income', ylabel='Density'>
```



```
dset[dset['Income'] > 600000]
```

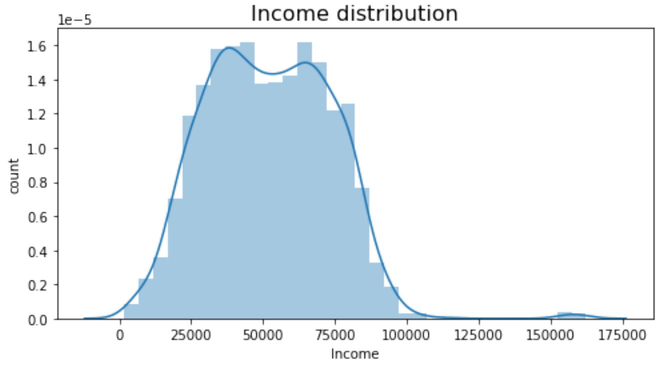
	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurchases
527	9432	1977	Graduation	Together	666666.0	1	0	2013-06-02	23	9	...	3

1 rows × 28 columns

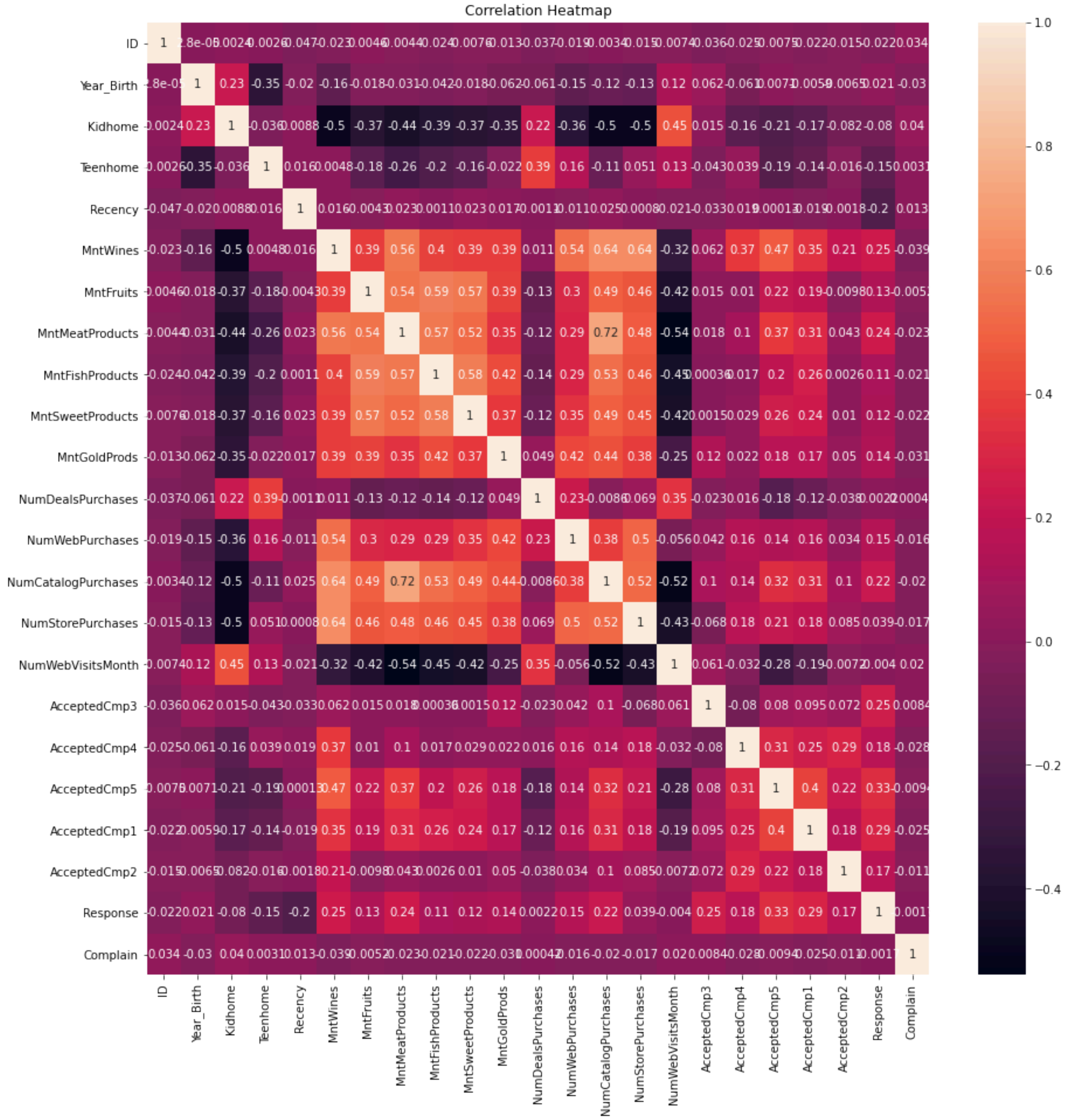
```
dset.drop(dset[dset['Income'] == 666666].index, inplace = True) #Drop one customer, who has 666666 $ income.
```

```
plt.figure(figsize=(8,4))
sns.distplot(dset['Income'], kde=True, hist=True)
plt.title('Income distribution', size=16)
plt.ylabel('count');
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and wi
  warnings.warn(msg, FutureWarning)
```



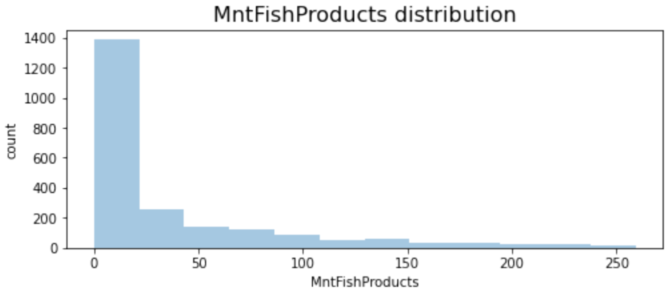
```
plt.figure(figsize=(15,15)) #Corellation Heatmap
plt.title(label = "Correlation Heatmap")
sns.heatmap(dset.corr(), annot=True)
plt.show()
```



Statistical Analysis

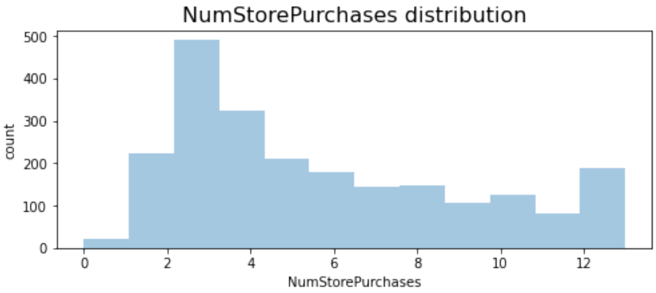
```
plt.figure(figsize=(8,3))
sns.distplot(dset['MntFishProducts'], kde=False, hist=True, bins=12)
plt.title('MntFishProducts distribution', size=16)
plt.ylabel('count');
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figur
```



```
plt.figure(figsize=(8,3))
sns.distplot(dset['NumStorePurchases'], kde=False, hist=True, bins=12)
```

```
plt.title('NumStorePurchases distribution', size=16)
plt.ylabel('count');
```



```
print("Total categories in the feature Education:\n\n")
print(dset["Education"].value_counts())
```

```
Total categories in the feature Education:


Graduation    1115
PhD            480
Master         365
2n Cycle       198
Basic           54
Name: Education, dtype: int64
```

```
fig = px.pie(dset, names='Education')
fig.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-e9ccfe306e6c> in <cell line: 1>()
----> 1 fig = px.pie(dset, names='Education')
      2 fig.show()

NameError: name 'px' is not defined
```

Next steps: Explain error

```
print("Total categories in the feature Marital_Status:\n")
print(dset["Marital_Status"].value_counts(), "\n")
```

```
Total categories in the feature Marital_Status:

Married     864
Together    580
Single      480
Divorced    232
Widow        77
Name: Marital_Status, dtype: int64
```

```
dset.drop(dset[dset['Marital_Status'] == "YOLO"].index, inplace = True)
dset.drop(dset[dset['Marital_Status'] == "Absurd"].index, inplace = True)
dset.drop(dset[dset['Marital_Status'] == "Alone"].index, inplace = True)
```

```
fig = px.pie(dset, names='Marital_Status')
fig.show()
```

```
# fig = px.histogram(dset,  x=" Income", y="Education", title="Income by Education")
# fig.show()
```

```
dset["Dt_Customer"][dset["Response"] == 1].dt.month.value_counts()
 #Count month when customers enroll with the company
```

```
8     46
9     41
10    38
11    33
1     30
2     28
3     26
5     24
4     22
12    18
6     16
7     11
Name: Dt_Customer, dtype: int64
```

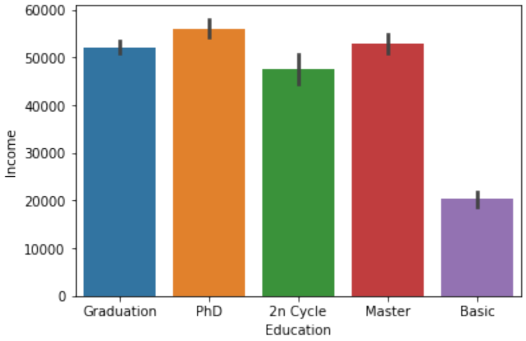
```
dset["Dt_Customer"][dset["Response"] == 1].dt.weekday.value_counts()
#Count days of week when customers enroll with the company
```

```
0    59
2    55
4    51
1    50
5    46
3    42
6    30
Name: Dt_Customer, dtype: int64
```

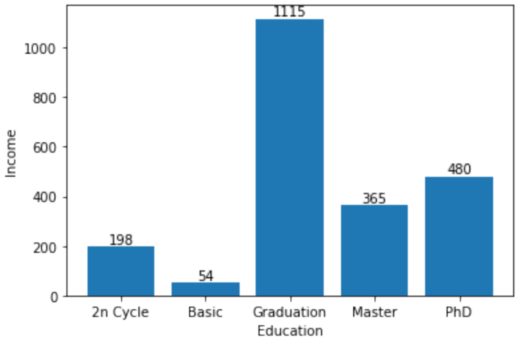
```
fig = px.histogram(dset, x="Country", y="TotalMnt", title="Total Amount Spent by Country")
# fig.title('Total Amount Spent by Country', size=16)
# fig.ylabel('Amount Spent');
fig.show()
```

```
sns.barplot(x=dset["Education"],y=dset["Income"])
```

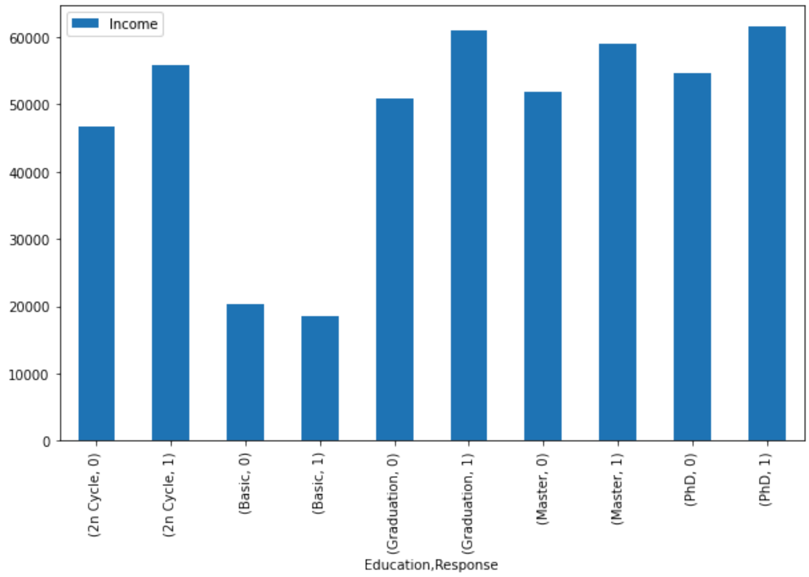
```
<AxesSubplot: xlabel='Education', ylabel='Income'>
```



```
plot = plt.bar(dset.groupby(by = "Education").count()["ID"].index.to_list(),
                dset.groupby(by = "Education").count()["ID"].to_list())

for value in plot:  #Add the data value on head of the bar
    height = value.get_height()
    plt.text(value.get_x() + value.get_width()/2.,
             1.002*height,'%d' % int(height), ha='center', va='bottom')

plt.xlabel("Education")
#Add labels and title
plt.ylabel("Income")

plt.rcParams["figure.figsize"] = (10,6)
#Change figure size for better visibility
plt.show()
```



```
dset.pivot_table(["Income"], ["Education","Response"], aggfunc="mean").plot.bar()
plt.show()
```

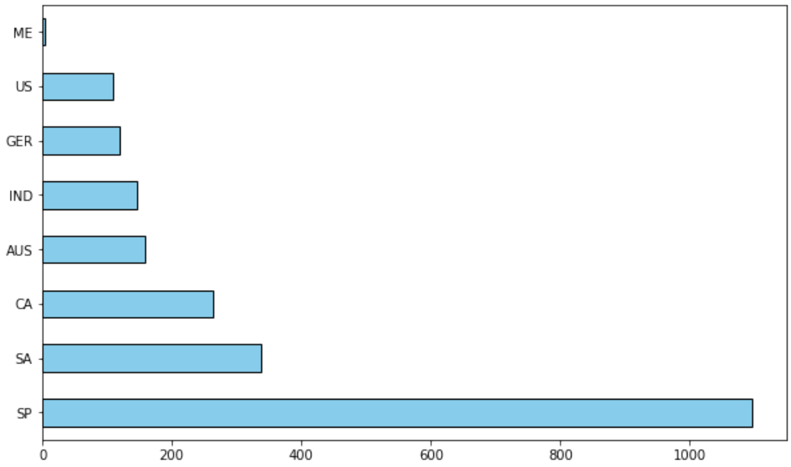


```
print("Total categories in the feature Countries:\n")
print(dset["Country"].value_counts(), "\n")
```

```
Total categories in the feature Countries:

SP     1092
SA      335
CA      261
AUS     146
IND     145
GER     116
US      107
ME        3
Name: Country, dtype: int64
```

```
dset['Country'].value_counts().plot(kind='barh', color='skyblue', edgecolor=(0,0,0))
```

```
<AxesSubplot: >
```

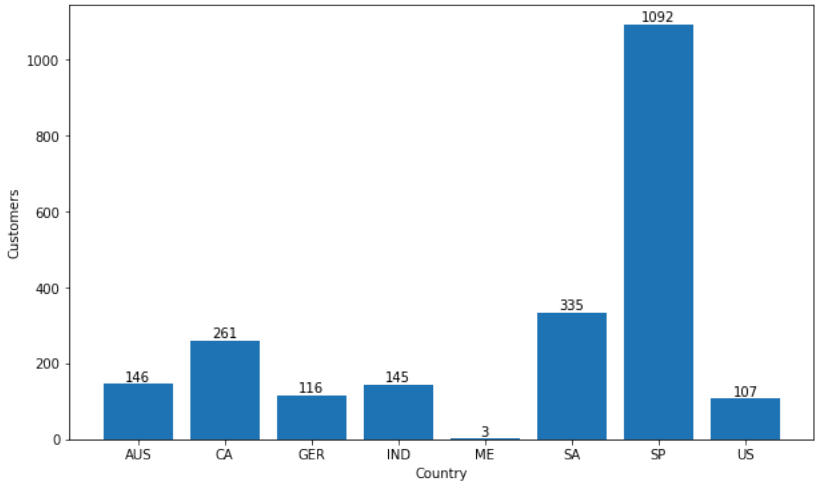


```
# plt.figure(figsize=(5,4))
# dset.groupby('Country')['TotalPurchases'].sum().sort_values(ascending=False).plot(kind='bar')
# plt.title('Total Number of Purchases by Country', size=16)
# plt.ylabel('Number of Purchases');

fig = px.histogram(dset, x="Country", y="TotalPurchases", color="Country")
fig.show()
```

```
dset["Marital_Status"][dset["Response"] == 1].value_counts()
#Count Marital_Status of customers of those who accepted last campaign
```

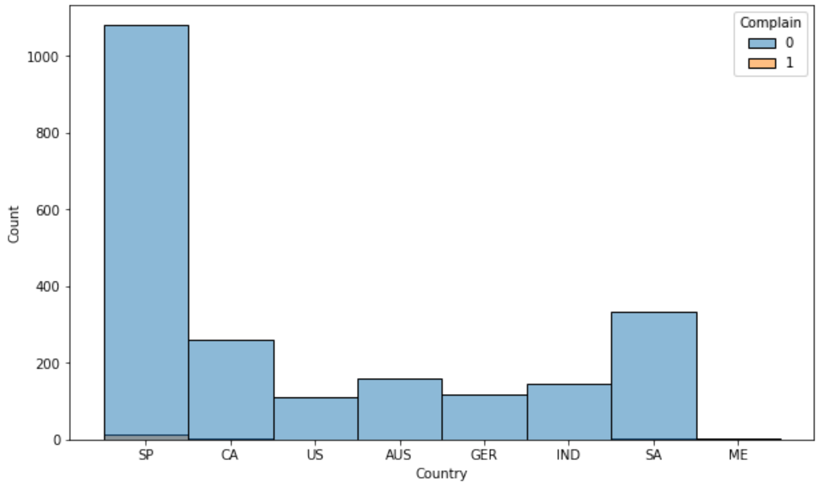
```
Single      106
Married      98
Together     60
Divorced     48
Widow        18
Name: Marital_Status, dtype: int64
```

```
plot = plt.bar(dset.groupby(by = "Country").count()["ID"].index.to_list(),
               dset.groupby(by = "Country").count()["ID"].to_list())

for value in plot:  #Add the data value on head of the bar
    height = value.get_height()
    plt.text(value.get_x() + value.get_width()/2.,
             1.002*height,'%d' % int(height), ha='center', va='bottom')

plt.xlabel("Country")  #Add labels and title
plt.ylabel("Customers")

plt.rcParams["figure.figsize"] = (10,6) #Change figure size for better visibility
plt.show()
```

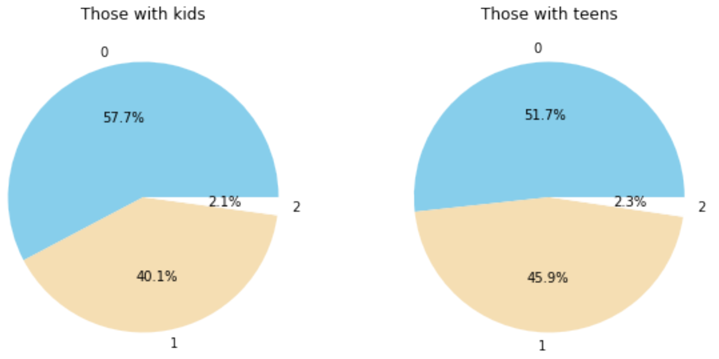


```
sns.histplot(data=dset, x="Country", hue="Complain", multiple="stack")
```

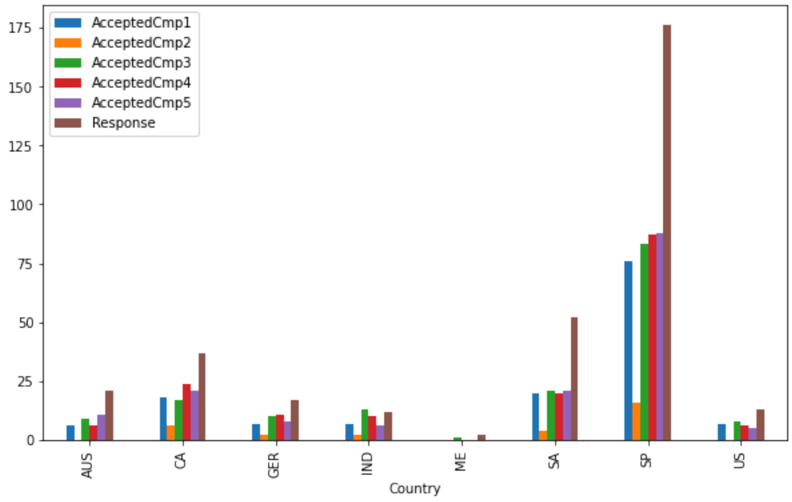
```
<AxesSubplot: xlabel='Country', ylabel='Count'>
```



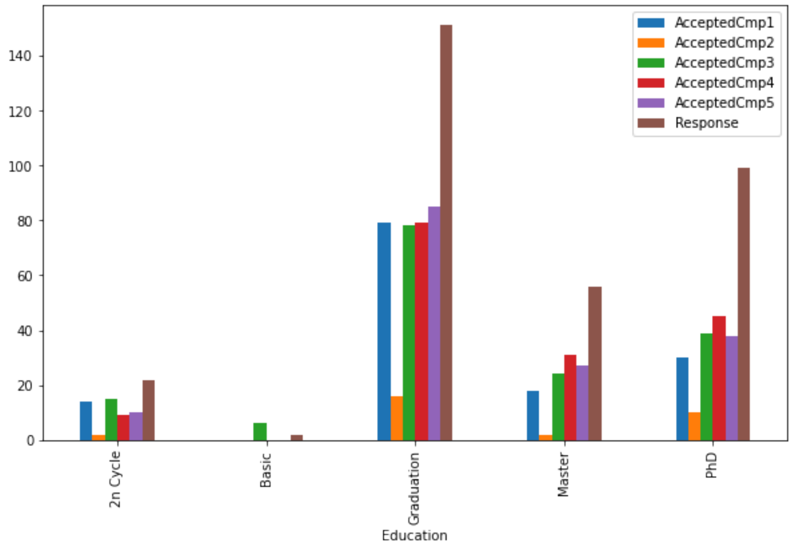
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(10,10)) #ax1,ax2 refer to your two pies

# 1,2 denotes 1 row, 2 columns - if you want to stack vertically, it would be 2,1
colors= 'skyblue', 'wheat', 'white'

labels = dset['Kidhome'].unique()
values = dset['Kidhome'].value_counts()
ax1.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%') #plot first pie


labels = dset['Teenhome'].unique()
values = dset['Teenhome'].value_counts()
ax2.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%') #plot second pie

ax1.set(aspect="equal", title='Those with kids')
ax2.set(aspect="equal", title='Those with teens')
plt.show()
```



```
dset.pivot_table(["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "Response"], ["Country"], aggfunc="sur
plt.show()
```

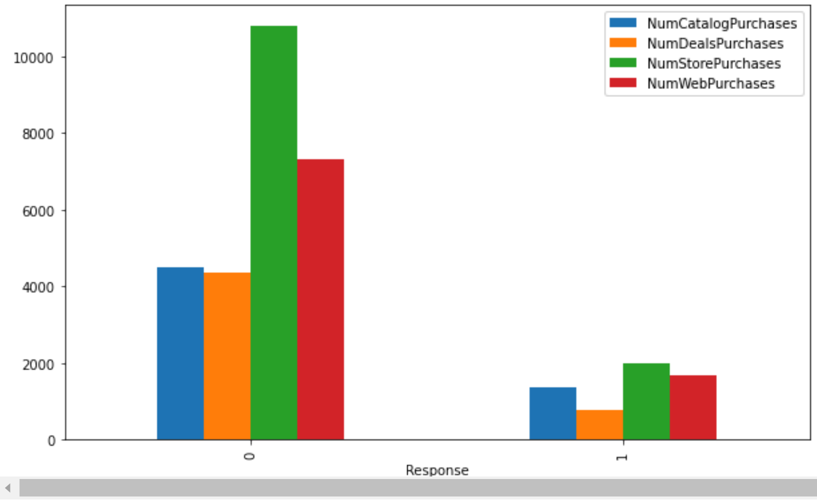


```
dset.pivot_table(["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "Response"], ["Education"], aggfunc="
plt.show()
```

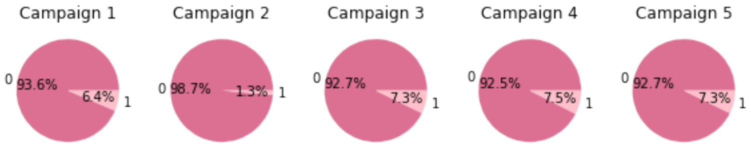


```
# convert country codes to correct nomenclature for choropleth plot
# the dataset doesn't provide information about country codes
## ...so I'm taking my best guess about the largest nations that make sense given the codes provided
dset['Country_code'] = dset['Country'].replace({'SP': 'ESP', 'CA': 'CAN', 'US': 'USA', 'SA': 'ZAF', 'ME': 'MEX'})

# success of campaigns by country code
df_cam = dset[['Country_code', 'AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Response']].melt(
    id_vars='Country_code', var_name='Campaign', value_name='Accepted (%)')
df_cam = pd.DataFrame(df_cam.groupby(['Country_code', 'Campaign'])['Accepted (%)'].mean()*100).reset_index(drop=False)

# rename the campaign variables so they're easier to interpret
df_cam['Campaign'] = df_cam['Campaign'].replace({'AcceptedCmp1': '1',
                                                'AcceptedCmp2': '2',
                                                'AcceptedCmp3': '3',
                                                'AcceptedCmp4': '4',
                                                'AcceptedCmp5': '5',
                                                 'Response': 'Most recent'
                                                })

# choropleth plot
import plotly.express as px

fig = px.choropleth(df_cam, locationmode='ISO-3', color='Accepted (%)', facet_col='Campaign', facet_col_wrap=2,
                    facet_row_spacing=0.05, facet_col_spacing=0.01, width=700,
                    locations='Country_code', projection='natural earth', title='Advertising Campaign Success Rate by Country'
                   )
fig.show()
```

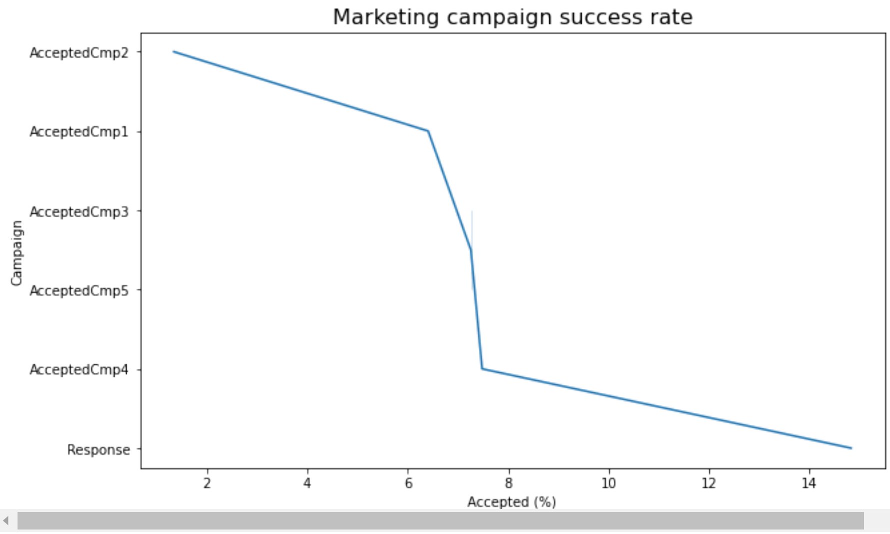
```
dset.pivot_table(["NumDealsPurchases","NumWebPurchases","NumCatalogPurchases","NumStorePurchases"], ["Response"], aggfunc="sum").plot.ba
plt.show()
```



```
fig, (ax1,ax2, ax3, ax4, ax5) = plt.subplots(1,5, figsize=(10,10))
# 1,5 denotes 1 row, 5 columns

colors= 'palevioletred', 'pink'

labels = dset['AcceptedCmp1'].unique()
values =  dset['AcceptedCmp1'].value_counts()
ax1.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%')

labels = dset['AcceptedCmp2'].unique()
values = dset['AcceptedCmp2'].value_counts()
ax2.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%')

labels = dset['AcceptedCmp3'].unique()
values = dset['AcceptedCmp3'].value_counts()
ax3.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%')

labels = dset['AcceptedCmp4'].unique()
values = dset['AcceptedCmp4'].value_counts()
ax4.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%')

labels = dset['AcceptedCmp5'].unique()
values = dset['AcceptedCmp5'].value_counts()
ax5.pie(values,labels = labels,colors = colors,autopct = '%1.1f%%')

ax1.set(aspect="equal", title='Campaign 1')
ax2.set(aspect="equal", title='Campaign 2')
ax3.set(aspect="equal", title='Campaign 3')
ax4.set(aspect="equal", title='Campaign 4')
ax5.set(aspect="equal", title='Campaign 5')
plt.show()
```



```
# calculate success rate (percent accepted)
cam_success = pd.DataFrame(dset[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Response']].mean()*10
                           columns=['Percent']).reset_index()

# plot
sns.lineplot(x='Percent', y='index', data=cam_success.sort_values('Percent'), palette='Blues')
plt.xlabel('Accepted (%)')
plt.ylabel('Campaign')
plt.title('Marketing campaign success rate', size=16);
```

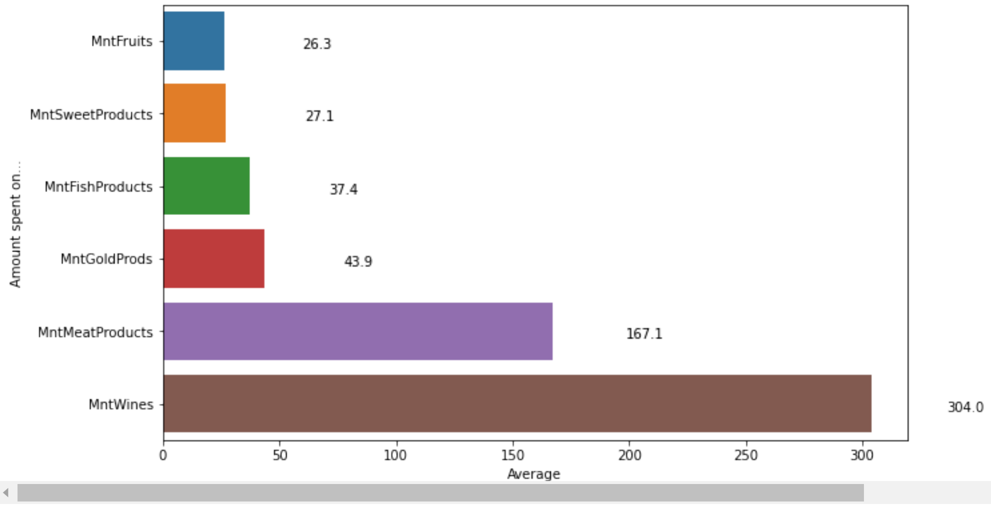


```
# list of cols with binary responses
binary_cols = [col for col in dset.columns if 'Accepted' in col] + ['Response', 'Complain']

# list of cols for spending
mnt_cols = [col for col in dset.columns if 'Mnt' in col]

# list of cols for channels
channel_cols = [col for col in dset.columns if 'Num' in col] + ['TotalPurchases', 'TotalCampaignsAcc']
```

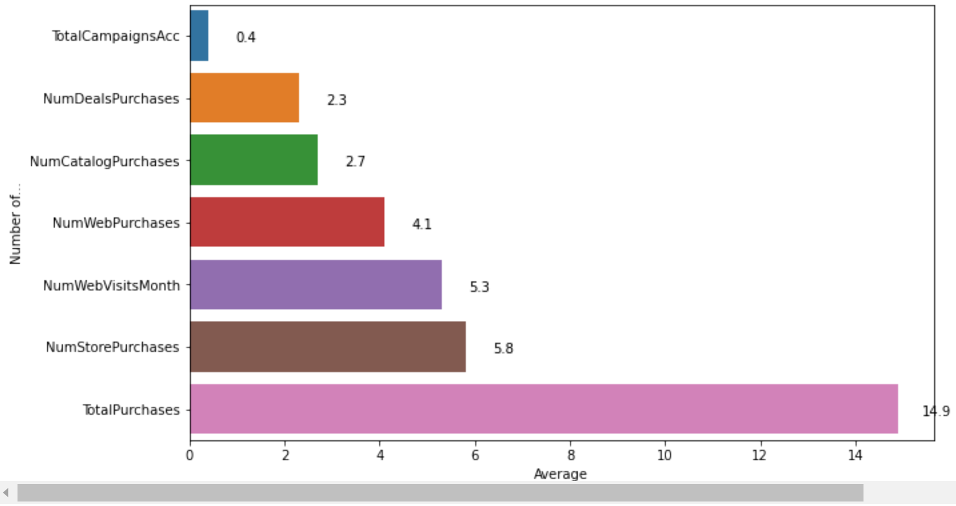
```
# average customer demographics
demographics = pd.DataFrame(round(dset.drop(columns=binary_cols+mnt_cols+channel_cols).mean(), 1), columns=['Average']).reindex([
    'Year_Birth', 'Year_Customer', 'Income', 'Dependents', 'Kidhome', 'Teenhome', 'Recency'])

demographics
```

	Average
Year_Birth	1968.9
Year_Customer	2013.0
Income	51954.6
Dependents	0.9
Kidhome	0.4
Teenhome	0.5
Recency	49.1

```
spending = pd.DataFrame(round(dset[mnt_cols].mean(), 1), columns=['Average']).sort_values(by='Average').reset_index()

# plot
ax = sns.barplot(x='Average', y='index', data=spending)
plt.ylabel('Amount spent on...')

## add text labels for each bar's value
for p,q in zip(ax.patches, spending['Average']):
    ax.text(x=q+40,
            y=p.get_y()+0.5,
            s=q,
            ha="center") ;
```



```
channels = pd.DataFrame(round(dset[channel_cols].mean(), 1), columns=['Average']).sort_values(by='Average').reset_index()

# plot
ax = sns.barplot(x='Average', y='index', data=channels)
plt.ylabel('Number of...')

## add text labels for each bar's value
for p,q in zip(ax.patches, channels['Average']):
    ax.text(x=q+0.8,
            y=p.get_y()+0.5,
            s=q,
            ha="center") ;
```

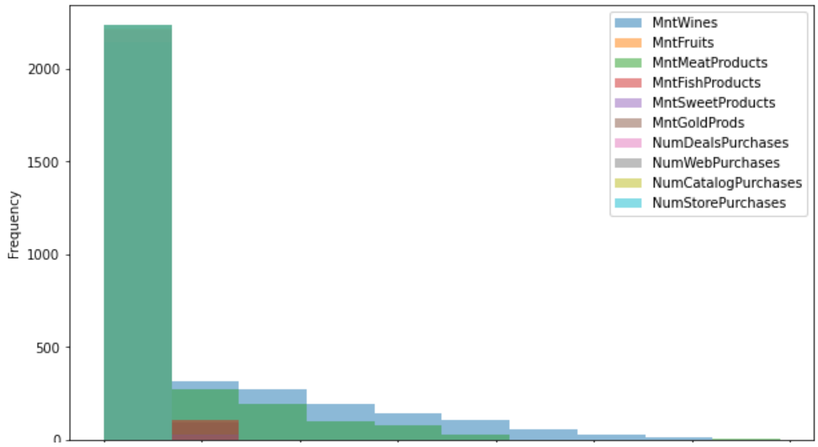


```
g = sns.FacetGrid(df_cam2, col='Campaign', col_wrap=3)
g.map(sns.barplot, 'Country', 'Accepted (%)')
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\axisgrid.py:645: UserWarning:

Using the barplot function without specifying `order` is likely to produce an incorrect plot.
```

```
#compare countries
countries = dset[['Country', 'MntWines','MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases' ]]

countries.plot.hist(alpha=0.5)
```

```
<AxesSubplot: ylabel='Frequency'>
```


Frequency
MntWines
MntFruits
MntMeatProducts
MntFishProducts
MntSweetProducts
MntGoldProds
NumDealsPurchases
NumWebPurchases
NumCatalogPurchases
NumStorePurchases