Market Analysis

Task Details

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

Section 01: Exploratory Data Analysis

- **Are there any null values or outliers? How will you wrangle/handle them?**
- **Are there any variables that warrant transformations?**
- **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?**

In [4]:

```
#Load libraries

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

#Load data set
data=pd.read_csv('/content/marketing_data.csv')
data.head()
```

Out[4]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFrui
0	1826	1970	Graduation	Divorced	$84,835.00	0	0	6/16/14	0	189	10
1	1	1961	Graduation	Single	$57,091.00	0	0	6/15/14	0	464	
2	10476	1958	Graduation	Married	$67,267.00	0	1	5/13/14	0	134	1
3	1386	1967	Graduation	Together	$32,474.00	1	1	5/11/14	0	10	
4	5371	1989	Graduation	Single	$21,474.00	1	0	4/8/14	0	6	1

In [5]:

```
#Detemine number of rows and columns
data.shape
```

Out[5]:

```
(2240, 28)
```

In [6]:

```
#Determine data summary
data.describe()
```

Out[6]:

	ID	Year_Birth	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFi

	ID	Year_Birth	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFi
count	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	
std	3246.662198	11.984069	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	
min	0.000000	1893.000000	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	
50%	5458.500000	1970.000000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	
75%	8427.750000	1977.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	
max	11191.000000	1996.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	

In [7]:

```
#Check featues, data types and null values
data.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
```

Observations:

- **Column 'Income' with space at the begining.**
- **Column 'Income' with null values.**
- **Column 'Income' with data type 'object'.**
- **Column 'Income' with missing values.**

Next Step:

- Remove space from column name 'Income'

- Remove space from column name 'Income'.
- **Convert Income data type to 'float'.**
- **Imputation of null values for feature 'Income'.**

In [8]:

```
#Remove space from column name 'Income'.
data.columns=data.columns.str.replace(" ","")
data.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 [9]:

```
#Delete $
data['Income']=data['Income'].str.replace('$','')

#Delete ','
data['Income']=data['Income'].str.replace(',','')

#Convert 'Income' data type to 'float'.
data['Income']=data['Income'].astype('float')
```

In [10]:

```
data.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
```

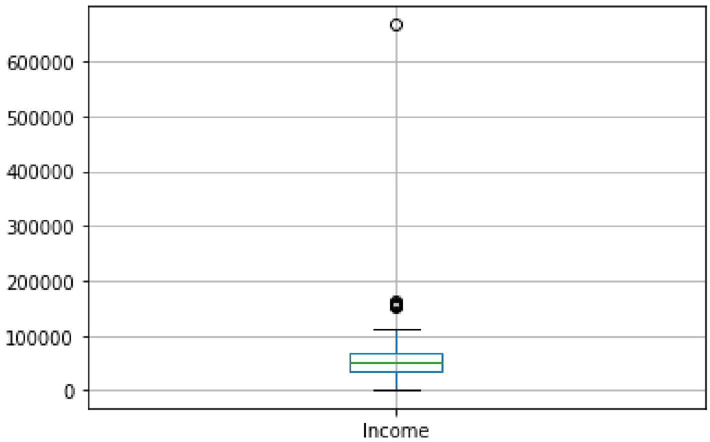
```
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   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: float64(1), int64(23), object(4)
memory usage: 490.1+ KB
```

Observation:

Null values in feature 'Income'

In [11]:

```
data.Income.plot(kind='box')
plt.grid()
```



Observation:

There are outliers in feature 'Income', some customers have very high income which are most likely natural outliers. So we will impute null values with median to minimize the effect the outliers.

In [12]:

```
data['Income']=data['Income'].fillna(data.Income.median())
```

In [13]:

```
data.isnull().sum()
```

Out[13]:

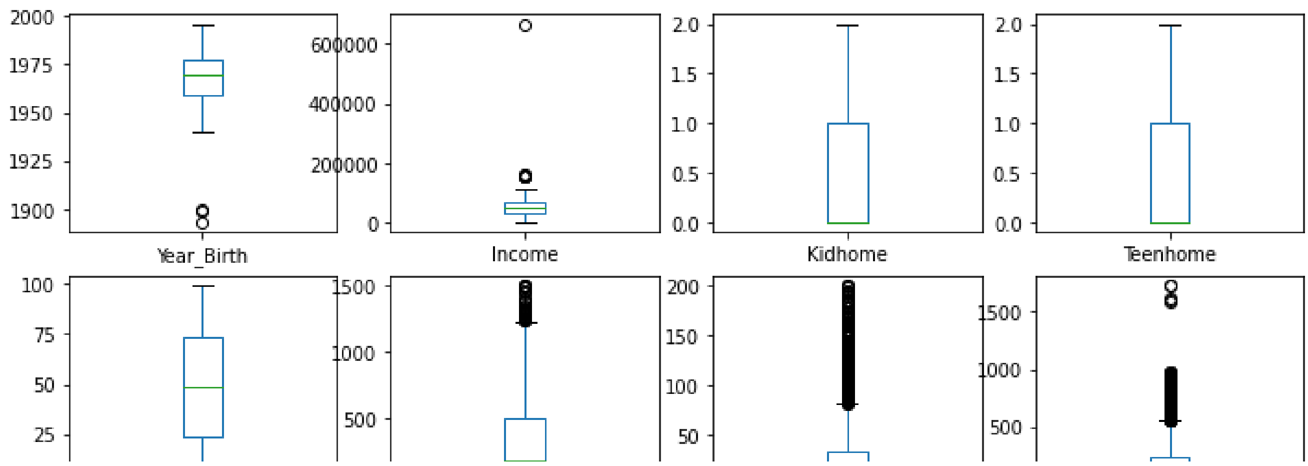
```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                  0
Kidhome                 0
Teenhome                0
Dt_Customer             0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         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
Complain                0
Country                 0
dtype: int64
```

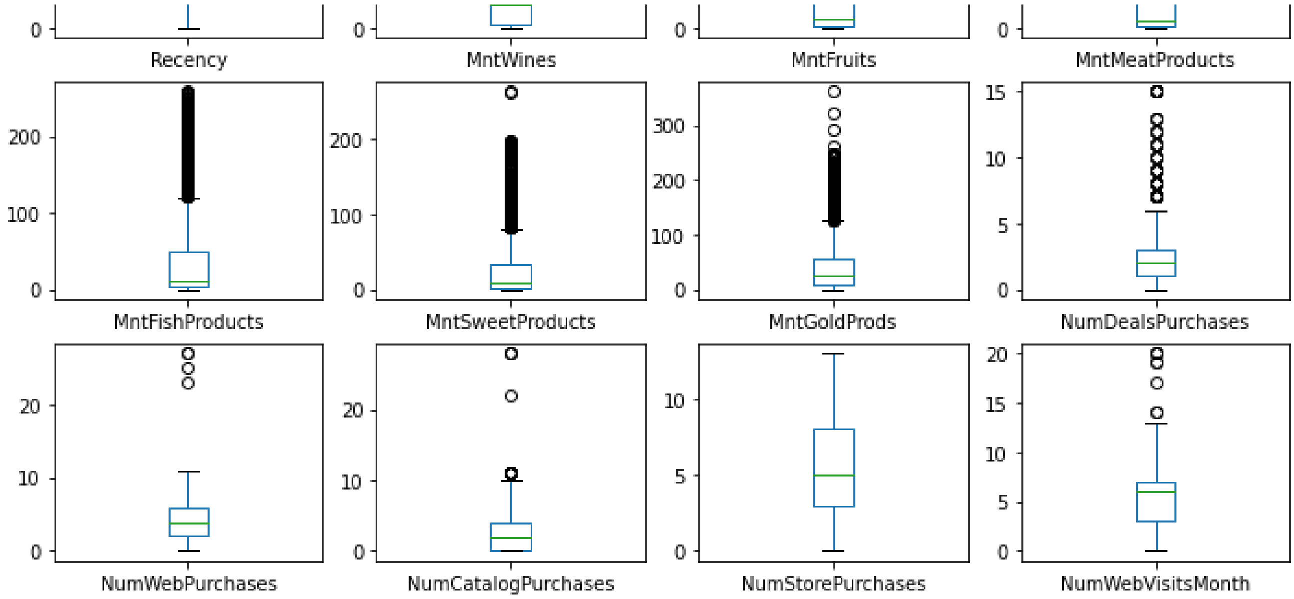
In [14]:

```
#Plot numerical variables
data_plot=data.drop(['ID', 'Education', 'Marital_Status', 'AcceptedCmp1', 'AcceptedCmp2',
'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Response', 'Complain'], axis=1)
data_plot.plot(subplots=True, layout=(4,4), kind='box', figsize=(12,10))
```

Out[14]:

```
Year_Birth                AxesSubplot(0.125,0.71587;0.168478x0.16413)
Income                 AxesSubplot(0.327174,0.71587;0.168478x0.16413)
Kidhome                AxesSubplot(0.529348,0.71587;0.168478x0.16413)
Teenhome               AxesSubplot(0.731522,0.71587;0.168478x0.16413)
Recency                  AxesSubplot(0.125,0.518913;0.168478x0.16413)
MntWines              AxesSubplot(0.327174,0.518913;0.168478x0.16413)
MntFruits             AxesSubplot(0.529348,0.518913;0.168478x0.16413)
MntMeatProducts       AxesSubplot(0.731522,0.518913;0.168478x0.16413)
MntFishProducts          AxesSubplot(0.125,0.321957;0.168478x0.16413)
MntSweetProducts      AxesSubplot(0.327174,0.321957;0.168478x0.16413)
MntGoldProds          AxesSubplot(0.529348,0.321957;0.168478x0.16413)
NumDealsPurchases     AxesSubplot(0.731522,0.321957;0.168478x0.16413)
NumWebPurchases             AxesSubplot(0.125,0.125;0.168478x0.16413)
NumCatalogPurchases      AxesSubplot(0.327174,0.125;0.168478x0.16413)
NumStorePurchases        AxesSubplot(0.529348,0.125;0.168478x0.16413)
NumWebVisitsMonth        AxesSubplot(0.731522,0.125;0.168478x0.16413)
dtype: object
```





Observation:

- **Outliers can be found in many cloumns, probably because of different buying behaviour.**
- **'Year_Birth' before 1900 is not possible.**

Next step:

- # Convert 'Year_Birth' to 'Age'.
- # Impute values of age >120.

In [15]:

```
#Converting birthdate to age
import datetime
now = datetime.datetime.now()
data['Age'] = now.year - data['Year_Birth']
```

In [16]:

```
data.head()
```

Out[16]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	0	189	104
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	0	464	5
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	0	134	11
3	1386	1967	Graduation	Together	32474.0	1	1	5/11/14	0	10	0
4	5371	1989	Graduation	Single	21474.0	1	0	4/8/14	0	6	16

In [17]:

```
#Checking correlation of other variables with 'Age'
data_corr = data.corr(method='kendall').unstack().sort_values(kind='quicksort', ascendin
g=False).reset_index()
data_corr.rename(columns={'level_0':'Variable 1','level_1':'Variable 2', 0:'Correlation c
oefficient'}, inplace=True)
data_corr[data_corr['Variable 1']=='Age']
```

Out[17]:

	Variable 1	Variable 2	Correlation coefficient
0	Age	Age	1.000000
127	Age	Teenhome	0.316054
195	Age	MntWines	0.161118
203	Age	Income	0.151713
216	Age	NumCatalogPurchases	0.131685
223	Age	NumStorePurchases	0.119001
227	Age	NumWebPurchases	0.116628
255	Age	MntMeatProducts	0.078941
266	Age	NumDealsPurchases	0.065301
269	Age	AcceptedCmp4	0.055132
275	Age	MntGoldProds	0.051854
306	Age	MntFishProducts	0.020495
318	Age	MntFruits	0.015737
323	Age	Recency	0.013955
331	Age	AcceptedCmp2	0.011130
350	Age	Complain	0.007115
360	Age	AcceptedCmp1	0.005108
384	Age	ID	-0.001701
394	Age	MntSweetProducts	-0.004110
419	Age	AcceptedCmp5	-0.012288
442	Age	Response	-0.017128
513	Age	AcceptedCmp3	-0.052968
537	Age	NumWebVisitsMonth	-0.096313
580	Age	Kidhome	-0.211408
623	Age	Year_Birth	-1.000000

Observation:</h1>

- **There no strong correlation of 'Age' with other variables, so we will replace 'Age'>120 will median value.**

In [18]:

```
data['Age'].median
```

Out[18]:

```
<bound method Series.median of 0        51
1        60
2        63
3        54
4        32
         ..
2235     45
2236     44
2237     45
2238     43
2239     52
Name: Age, Length: 2240, dtype: int64>
```

In [19]:

```
data['Age']= np.where(data['Age']>120, 51, data['Age'])
data.describe()
```

Out[19]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeat
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	224
mean	5592.159821	1968.805804	52237.975446	0.444196	0.506250	49.109375	303.935714	26.302232	16
std	3246.662198	11.984069	25037.955891	0.538398	0.544538	28.962453	336.597393	39.773434	22
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2828.250000	1959.000000	35538.750000	0.000000	0.000000	24.000000	23.750000	1.000000	1
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	6
75%	8427.750000	1977.000000	68289.750000	1.000000	1.000000	74.000000	504.250000	33.000000	23
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	172

Compressing the data by merging similar columns into one

- **Minors = Kidhome + Teenhome**

- **Total amount spent = Amount spent for wine + fruits + meat + fish + sweet + gold**
- **Number of all purchases = Purchases in store + catalog + web + deals**
- **Remote purchases = Purchases in catalog + web**
- **Marketing responsiveness = AcceptedCmp1 + AcceptedCmp2 + AcceptedCmp3 +AcceptedCmp4 + AcceptedCmp5 + Response**
-

Transformation:

- **Feature "Dt_customer" converted to "Customer_since"**
- **Delete "Year_birth", but keep "Age"**

In [20]:

```
#Minors in household
data['Minors']= data['Kidhome'] + data['Teenhome']

#Total amount spent
data['Amount_spent']= data['MntWines'] + data['MntFruits'] + data['MntMeatProducts'] + d
ata['MntFishProducts'] + data['MntSweetProducts'] + data['MntGoldProds']

#Amount spent on luxury items
data['Lux_spent']= data['MntGoldProds'] + data['MntWines']

#Number of total purchases
data['NumPur']= data['NumStorePurchases'] + data['NumCatalogPurchases'] + data['NumWebPur
chases'] + data['NumDealsPurchases']

#Number of remote purchases
data['RemPur']= data['NumCatalogPurchases'] + data['NumWebPurchases']

#Marketing responsiveness
data['Responsiveness']= data['AcceptedCmp1'] + data['AcceptedCmp2'] + data['AcceptedCmp3'
] + data['AcceptedCmp4'] + data['AcceptedCmp5']

#Convert 'Dt_customer' to 'Customer_since'
data['Customer_since']= pd.DatetimeIndex(data['Dt_Customer']).year
data=data.drop(['Dt_Customer'], axis=1)

#Drop 'Year_Birth'
data= data.drop(['Year_Birth'], axis=1)

data.head()
```

Out[20]:

	ID	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFis
0	1826	Graduation	Divorced	84835.0	0	0	0	189	104	379	
1	1	Graduation	Single	57091.0	0	0	0	464	5	64	
2	10476	Graduation	Married	67267.0	0	1	0	134	11	59	
3	1386	Graduation	Together	32474.0	1	1	0	10	0	1	
4	5371	Graduation	Single	21474.0	1	0	0	6	16	24	

In [21]:

```
data.columns
```

Out[21]:

```
Index(['ID', 'Education', 'Marital_Status', 'Income', 'Kidhome', 'Teenhome',
       'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',
       'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
       'Response', 'Complain', 'Country', 'Age', 'Minors', 'Amount_spent',
       'Lux_spent', 'NumPur', 'RemPur', 'Responsiveness', 'Customer_since'],
      dtype='object')
```

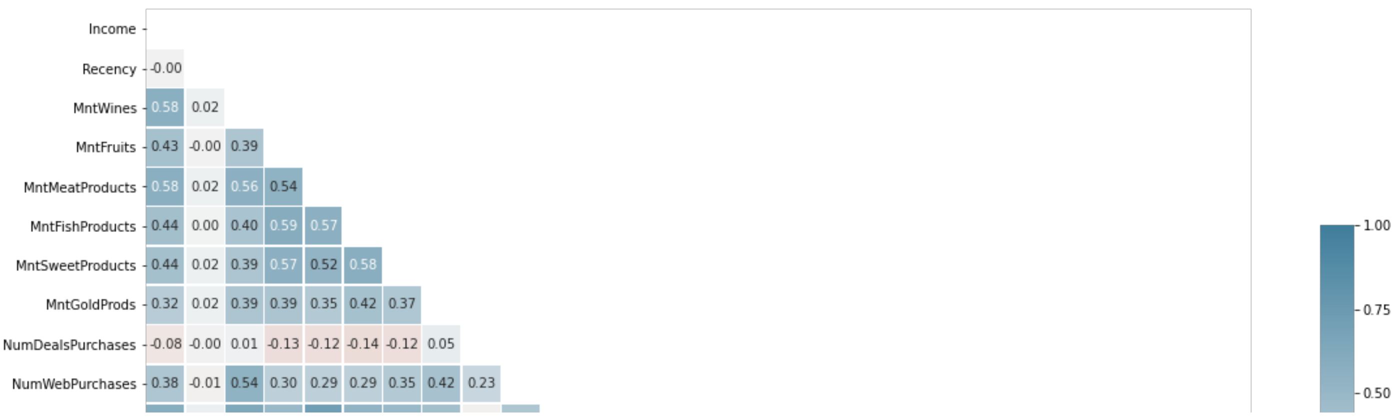
Explore dataset with correlation matrix

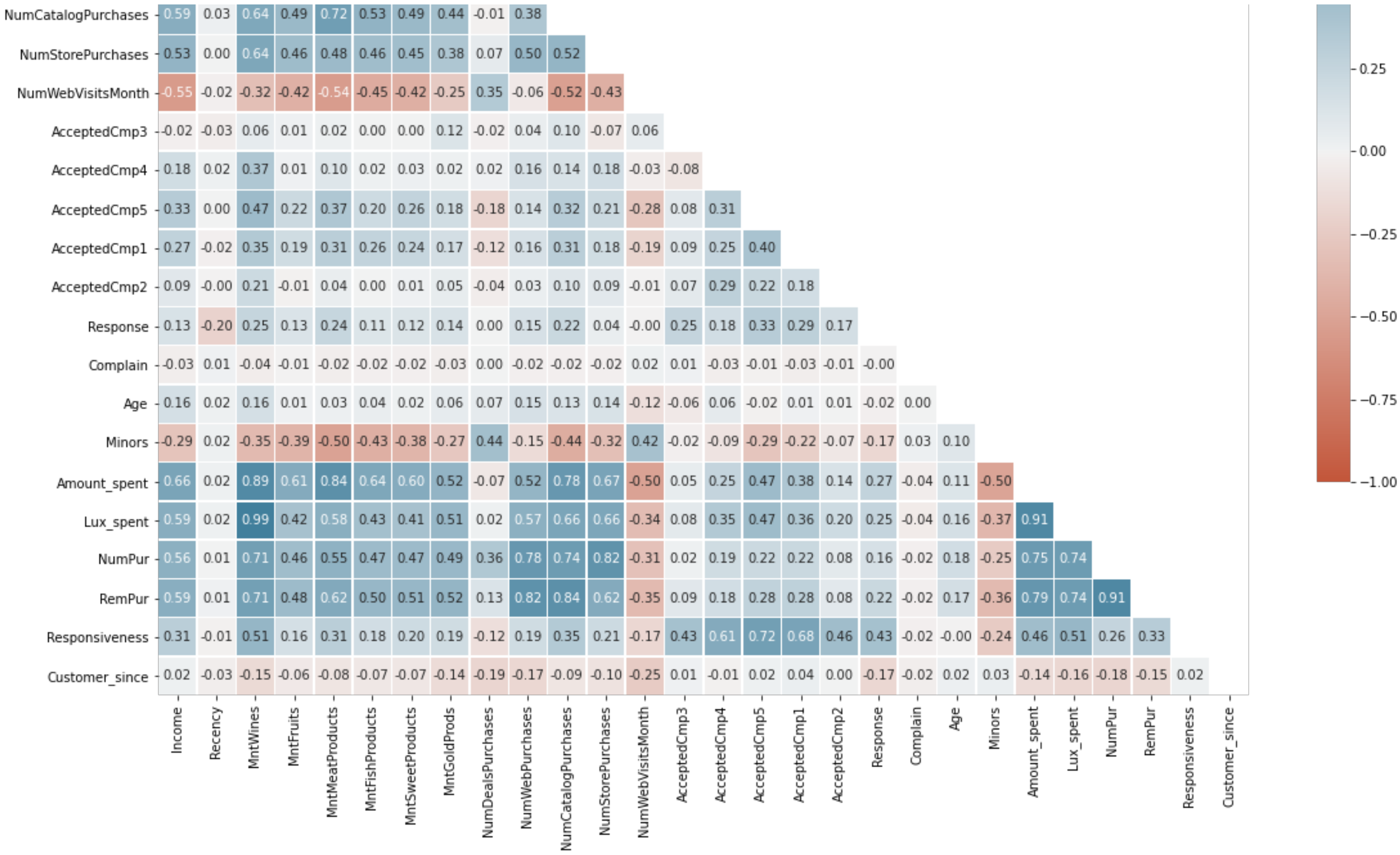
In [22]:

```
data_corr= data.drop(columns=['ID', 'Kidhome', 'Teenhome']).select_dtypes(include= np.nu
mber)

#Compute correlation matrix
corr= data_corr.corr()

#Generate a mask for upper triangle
mask= np.triu(np.ones_like(corr, dtype=bool))

#Set up matplotlib fig
f, ax= plt.subplots(figsize=(19, 19))

#Generate a custom diverging colormap
cmap = sns.diverging_palette(20, 230, as_cmap=True)

#Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmin= -1, vmax= 1, annot=True, fmt= '.2f', cente
r=0, square=True, linewidths= .5, cbar_kws={'shrink': .5})
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd4096ada10>
```





Explore effects of Income

In [23]:

```
data_corr= data.corr().unstack().sort_values(kind='quicksort', ascending=False).reset_index()
data_corr.rename(columns={'level_0': 'Column_1', 'level_1': 'Column_2', 0: 'Correlation Coefficient'}, inplace= True)
data_corr[data_corr['Column_1']== 'Income']
```

Out[23]:

	Column_1	Column_2	Correlation Coefficient
27	Income	Income	1.000000
78	Income	Amount_spent	0.664775
101	Income	NumCatalogPurchases	0.586826
104	Income	Lux_spent	0.585988
105	Income	RemPur	0.585698
109	Income	MntMeatProducts	0.577805
111	Income	MntWines	0.576903
122	Income	NumPur	0.563450
133	Income	NumStorePurchases	0.526600
196	Income	MntFishProducts	0.437564
198	Income	MntSweetProducts	0.436131
204	Income	MntFruits	0.428791
236	Income	NumWebPurchases	0.380554
263	Income	AcceptedCmp5	0.334893
272	Income	MntGoldProds	0.321938
279	Income	Responsiveness	0.307122
297	Income	AcceptedCmp1	0.274891

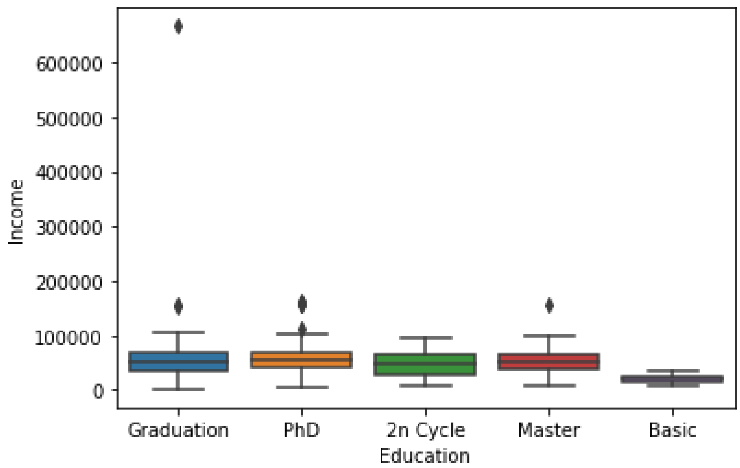
	Column_1	Column_2	Correlation Coefficient
360	Income	AcceptedCmp[illegible]	[illegible]
384	Income	Age	0.162232
416	Income	Response	0.132867
443	Income	AcceptedCmp2	0.087581
515	Income	Customer_since	0.022381
533	Income	Teenhome	0.018965
564	Income	ID	0.012996
634	Income	Recency	-0.004061
679	Income	AcceptedCmp3	-0.016064
736	Income	Complain	-0.027187
787	Income	NumDealsPurchases	-0.082315
887	Income	Minors	-0.290858
923	Income	Kidhome	-0.425326
958	Income	NumWebVisitsMonth	-0.549785

In [24]:

```
sns.boxplot(x="Education", y='Income', data=data)
```

Out[24]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fd3ff3ef350>

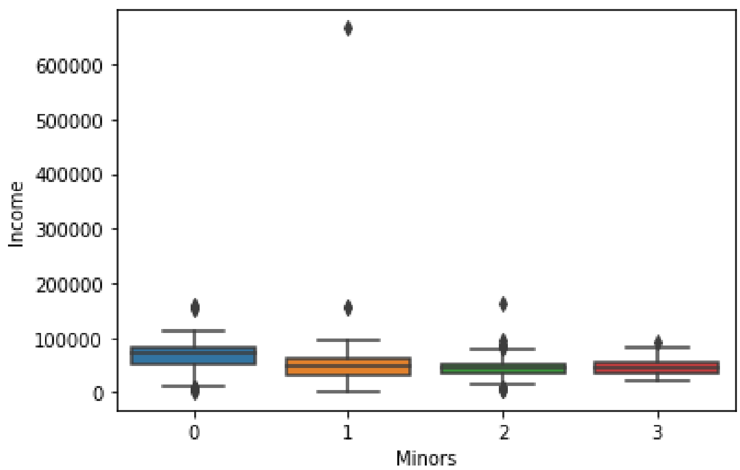


In [25]:

```
sns.boxplot(x= 'Minors', y='Income', data=data)
```

Out[25]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fd3ff3a6e50>



Observations

Income is highly corrrelated with:

- **Total Amount spent**
- **Catalog purchases**
- **Total amount spent on luxury items**
- **Total amount spent on remote purchases**
- **Amount spent for meat purchases**
- **Amount spent for wine purchases**
- **Number of purchases**
- **Number of store purchases**
- **Higher education (above basic)**

Income is negatively correlated with:

- **Monthly websote visits**
- **Presence of minors in household**

Effect of minor on other variables

In [26]:

```
data_corr= data.corr().unstack().sort_values(kind='quicksort', ascending=False).reset_in
dex()
data_corr.rename(columns={'level_0': 'Column_1', 'level_1': 'Column_2', 0: 'Correlation
Coefficient'}, inplace= True)
data_corr[data_corr['Column_1']== 'Minors']
```

Out[26]:

	Column_1	Column_2	Correlation Coefficient
5	Minors	Minors	1.000000
70	Minors	Teenhome	0.698433
71	Minors	Kidhome	0.689971
192	Minors	NumDealsPurchases	0.439684
214	Minors	NumWebVisitsMonth	0.418419
438	Minors	Age	0.095494
498	Minors	Customer_since	0.032215
500	Minors	Complain	0.031066
539	Minors	Recency	0.018053
617	Minors	ID	-0.000146
694	Minors	AcceptedCmp3	-0.020402
777	Minors	AcceptedCmp2	-0.069823
793	Minors	AcceptedCmp4	-0.087563
823	Minors	NumWebPurchases	-0.146361
842	Minors	Response	-0.169163
867	Minors	AcceptedCmp1	-0.224887
872	Minors	Responsiveness	-0.244282
874	Minors	NumPur	-0.245790
882	Minors	MntGoldProds	-0.266095

	Column_1	Column_2	Correlation Coefficient
886	Minors	AcceptedCmp5	-0.285642
888	Minors	Income	-0.290858
894	Minors	NumStorePurchases	-0.321125
900	Minors	MntWines	-0.351909
904	Minors	RemPur	-0.357523
908	Minors	Lux_spent	-0.367552
914	Minors	MntSweetProducts	-0.383137
918	Minors	MntFruits	-0.394853
926	Minors	MntFishProducts	-0.425503
931	Minors	NumCatalogPurchases	-0.439904
939	Minors	Amount_spent	-0.498888
945	Minors	MntMeatProducts	-0.502208

Observations:

Minor is positively correlated with:

- Number of deals purchased
- Number of web visits per month

The presence of Minors in household is negatively correlated with:

- Amount spent on meat purchases
- Total amount spent
- Number of catalogue purchases
- Amount spent on fish porducts
- Amount spent on Fruits
- Amount spent of sweets
- Luxury items
- Remote purchases
- Amount spent on wine purchases
- Number of store purchases
- Income

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?
- Does US fare significantly better than the Rest of the World in terms of total 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? What other factors are significantly related to amount spent on fish? (Hint: use your knowledge of interaction variables/effects).
- Is there a significant relationship between geographical regional and success of a campaign?

1. What factors are significantly related to the number of store purchases?

In [27]:

```
plt.figure(figsize=(8,3))
sns.distplot(data['NumStorePurchases'], kde=False, hist=True, bins=12)
plt.title('NumStorePurchases distribution', size=10)
plt.ylabel('count')
plt.grid()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

In [28]:

```
# Dropping 'ID' column since it is unique to every customer
data_reg= data.drop(['ID'], axis =1)
data_reg.head()
```

Out[28]:

	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProdu
0	Graduation	Divorced	84835.0	0	0	0	189	104	379	1
1	Graduation	Single	57091.0	0	0	0	464	5	64	
2	Graduation	Married	67267.0	0	1	0	134	11	59	
3	Graduation	Together	32474.0	1	1	0	10	0	1	
4	Graduation	Single	21474.0	1	0	0	6	16	24	

In [29]:

```
# Perform one-hot encoding of categorical variables (basically creating separate column f
or each category of a variable column)
def create_dummies(data,column_name):
  dummies = pd. get_dummies(data[column_name], prefix= column_name)
  data=pd.concat([data,dummies], axis=1)
  return data

data_reg= create_dummies(data_reg, 'Education')
data_reg= create_dummies(data_reg, 'Marital_Status')
data_reg= create_dummies(data_reg, 'Country')
data_reg.head()
```

Out[29]:

	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProdu
0	Graduation	Divorced	84835.0	0	0	0	189	104	379	1
1	Graduation	Single	57091.0	0	0	0	464	5	64	
2	Graduation	Married	67267.0	0	1	0	134	11	59	

	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProduc
3	Graduation	Together	32474.0	1	1	0	10	0	1	
4	Graduation	Single	21474.0	1	0	0	6	16	24	

In [30]:

```
data_reg.columns
```

Out[30]:

```
Index(['Education', 'Marital_Status', 'Income', 'Kidhome', 'Teenhome',
       'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',
       'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
       'Response', 'Complain', 'Country', 'Age', 'Minors', 'Amount_spent',
       'Lux_spent', 'NumPur', 'RemPur', 'Responsiveness', 'Customer_since',
       'Education_2n Cycle', 'Education_Basic', 'Education_Graduation',
       'Education_Master', 'Education_PhD', 'Marital_Status_Absurd',
       'Marital_Status_Alone', 'Marital_Status_Divorced',
       'Marital_Status_Married', 'Marital_Status_Single',
       'Marital_Status_Together', 'Marital_Status_Widow',
       'Marital_Status_YOLO', 'Country_AUS', 'Country_CA', 'Country_GER',
       'Country_IND', 'Country_ME', 'Country_SA', 'Country_SP', 'Country_US'],
      dtype='object')
```

In [31]:

```
# Dropping the categorical variables and NumStorePurchases and storing rest in a separate
data frame
data_reg_dropped= data_reg.drop(['Education', 'Marital_Status', 'Country', 'NumStorePurch
ases'], axis=1)
data_reg_dropped.head()
```

Out[31]:

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	Mnt
0	84835.0	0	0	0	189	104	379	111	189	
1	57091.0	0	0	0	464	5	64	7	0	
2	67267.0	0	1	0	134	11	59	15	2	
3	32474.0	1	1	0	10	0	1	0	0	
4	21474.0	1	0	0	6	16	24	11	0	

In [32]:

```
X= data_reg_dropped
y= data_reg['NumStorePurchases']
```

- **Fit linear regression model to training data (70% of dataset)**
- **Evaluate predictions on test data (30% of dataset).**

In [33]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
train_X, test_X, train_y, test_y = train_test_split(data_reg_dropped, y, test_size=0.3,r
andom_state=1)

#Liner Regression model
```

```
reg = LinearRegression(normalize=True)
reg.fit(train_X,train_y)

#Predictions
y_pred = reg.predict(test_X)


mean_absolute_error(test_y, y_pred)
```

Out[33]:

```
9.381714981601806e-15
```

Identify features that significantly affect the number of store purchases, using permutation importance:

In [34]:

```
!pip install eli5
```

```
Collecting eli5
  Downloading eli5-0.11.0-py2.py3-none-any.whl (106 kB)
     |████████████████████████████████| 106 kB 5.5 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from eli5
) (1.4.1)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.7/dist-packages
(from eli5) (0.8.9)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from eli
5) (2.11.3)
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.7/dist-packag
es (from eli5) (0.22.2.post1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from e
li5) (0.10.1)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.7/dist-packages (fr
om eli5) (1.19.5)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.7/dist-packages (fr
om eli5) (21.2.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from eli5)
(1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (fr
om scikit-learn>=0.20->eli5) (1.0.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages
(from jinja2->eli5) (2.0.1)
Installing collected packages: eli5
Successfully installed eli5-0.11.0
```

In [35]:

```
import eli5
from eli5.sklearn import PermutationImportance

perm=PermutationImportance(reg, random_state=0).fit(test_X, test_y)
eli5.show_weights(perm, feature_names= test_X.columns.tolist(), top=10)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: T
he sklearn.feature_selection.base module is  deprecated in version 0.22 and will be remov
ed in version 0.24. The corresponding classes / functions should instead be imported from
sklearn.feature_selection. Anything that cannot be imported from sklearn.feature_selectio
n is now part of the private API.
  warnings.warn(message, FutureWarning)
```

Out[35]:

Weight	Feature
11.7184 ± 0.9437	NumPur
0.8250 ± 0.0583	RemPur
0.6097 ± 0.0352	NumDealsPurchases
0.5911 ± 0.0378	NumCatalogPurchases
0.5270 ± 0.0291	NumWebPurchases
0.0021 ± 0.0001	Amount_spent
0.0007 ± 0.0000	Lux_spent
0.0004 ± 0.0000	Education_Graduation
0.0003 ± 0.0000	MntMeatProducts
0.0003 ± 0.0000	Education_PhD

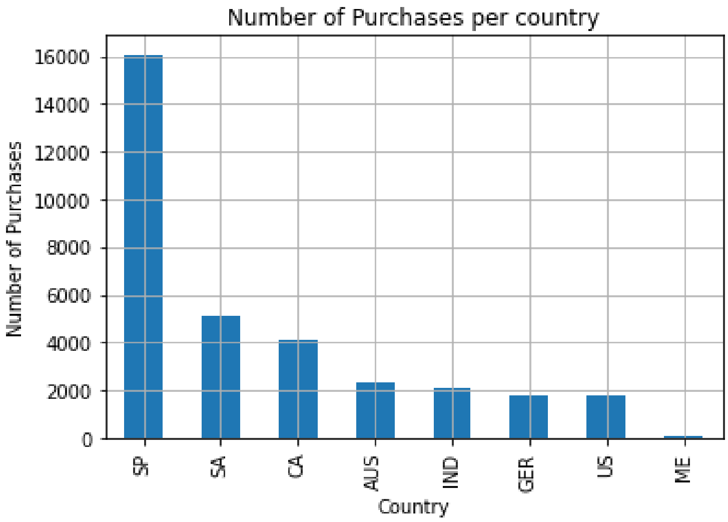
Weight Feature

Most important features of Store purchases are Remore purchases, Number of deals purchased, Number of web purchases, Number of catalogue purchases.

2. Does US fare significantly better than the Rest of the World in terms of total purchases? **

In [36]:

```
plt.figure()
data.groupby('Country')['NumPur'].sum().sort_values(ascending=False).plot(kind='bar')
plt.title('Number of Purchases per country')
plt.ylabel('Number of Purchases')
plt.grid()
```



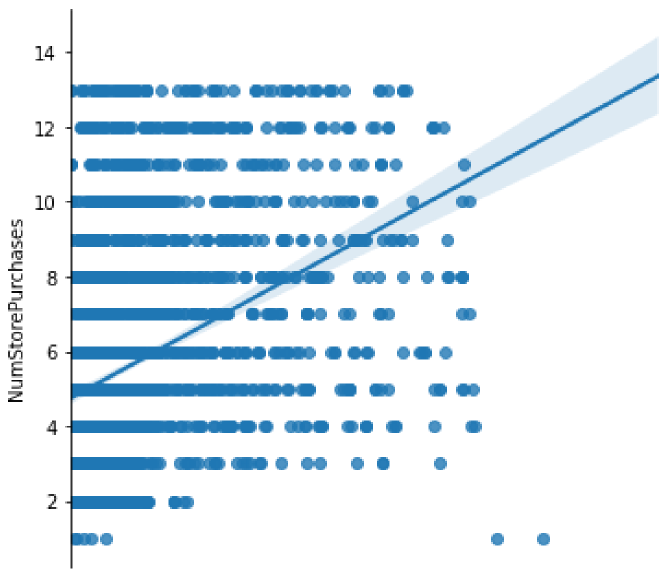
US does not fare better in Total Purchases. Spain, South Africa, Canada, Australia and India have higher number of purchases than US.

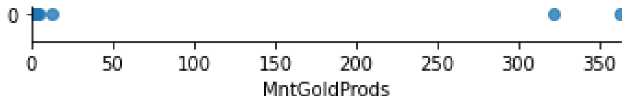
3. Your supervisor insists that people who buy gold are more conservative. Therefore, people who spent an above average amount on gold in the last 2 years would have more in store purchases. Justify or refute this statement using an appropriate statistical test

- **Plot relationship between amount spent in gold in last 2 years (MntGoldProds) and Number of in store purchases.**

In [37]:

```
sns.lmplot(x= 'MntGoldProds', y='NumStorePurchases', data= data);
```





Findings: There is a positive relationship, but is it significant?

In [38]:

```
!pip install scipy
```

```
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (f
rom scipy) (1.19.5)
```

In [39]:

```
from scipy import stats
tau, p_value = stats.kendalltau(data['MntGoldProds'], data['NumStorePurchases'])
p_value
```

Out[39]:

```
4.752746314649227e-152
```

Findings: People who spent an above average amount on gold have indeed more in store purchases. This correlation is statistically significant, however, this does not prove causation that people who spent money on gold are more conservative and prefer buying in stores.

4. 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?

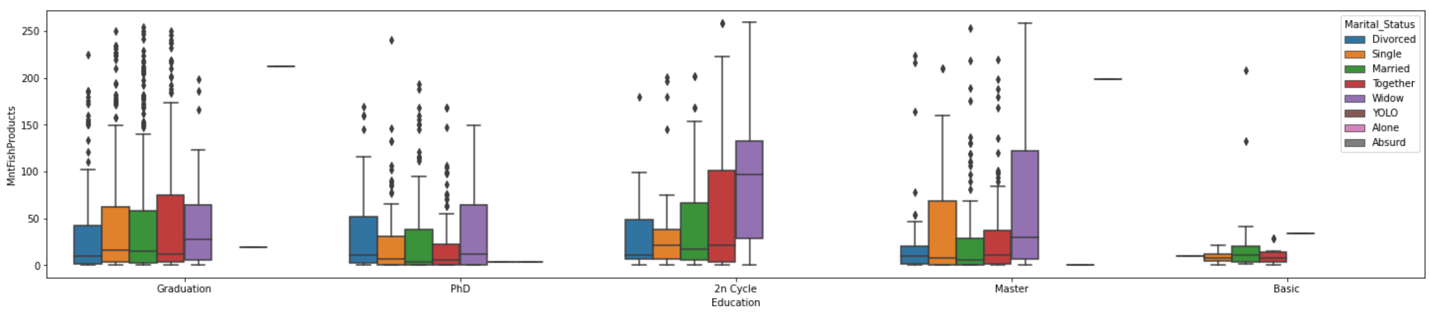
We will compare MntFishProducts between Married PhD candidates and all other customers:

In [40]:

```
plt.figure(figsize=(25,5))
sns.boxplot(x= data['Education'], y= data['MntFishProducts'], hue= data['Marital_Status'
])
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3adcb1b50>
```



Findings: Ph.d Married people do not spend more on Fish Products.

Now to find out what other factors are significantly related to amount spent of Fish:

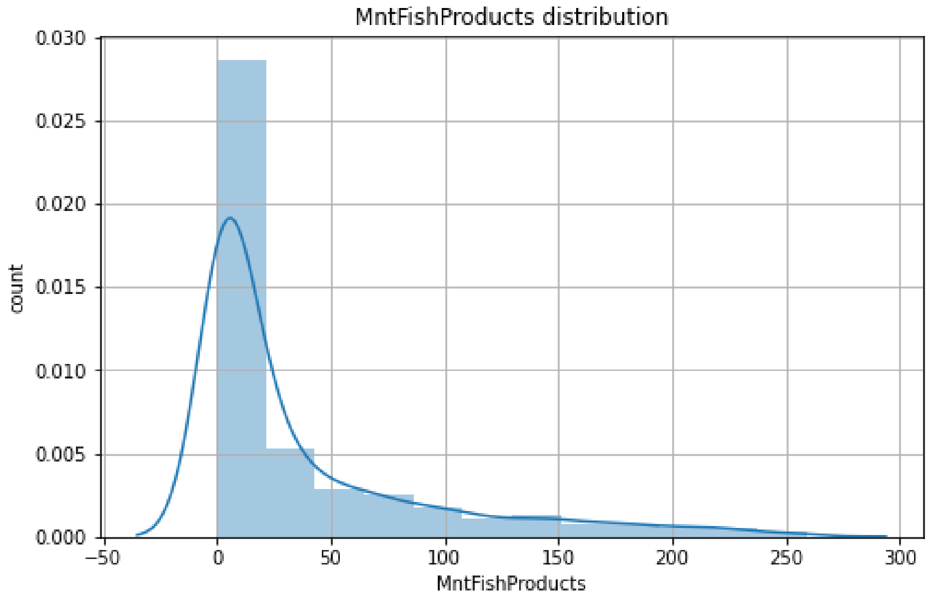
- **Like with the analysis of NumStorePurchases above, we will use use a linear regression model with MntFishProducts as the target variable, and then use machine learning explainability techniques to get insights about which features predict the amount spent on fish**
- **Begin by plotting the target variable:**

In [41]:

```
plt.figure(figsize=(8,5))
```

```
sns.distplot(data['MntFishProducts'], kde= 'False', hist=True, bins=12)
plt.title('MntFishProducts distribution')
plt.ylabel('count')
plt.grid()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [42]:

```
#  now drop categorical columns like we did before for applying regression model
X= data_reg_dropped2= data_reg.drop(['Education', 'Marital_Status', 'Country', 'MntFishP
roducts'], axis=1)
y= data_reg_dropped['MntFishProducts']
```

- **Fit linear regression model to training data (70% of dataset)**
- **Evaluate predictions on test data (30% of dataset)**

In [43]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

train_X, test_X, train_y, test_y= train_test_split(data_reg_dropped2, y, test_size= 0.2,
random_state=42)

#Linear Regression model
reg= LinearRegression(normalize=True)
reg.fit(train_X, train_y)

#prediction
y_pred= reg.predict(test_X)
mean_absolute_error(test_y, y_pred)
```

Out[43]:

```
5.026247275528663e-13
```

Identify features that significantly affect the amount spent on fish, using permutation importance:

In [44]:

```
import eli5
from eli5.sklearn import permutation_importance
```

```
perm= PermutationImportance(reg, random_state=0).fit(test_X, test_y)
eli5.show_weights(perm, feature_names = test_X.columns.tolist(), top=10)
```

Out[44]:

Weight	Feature
321.0849 ± 22.6471	Amount_spent
42.3584 ± 1.8337	MntMeatProducts
27.4869 ± 2.4586	MntWines
27.2073 ± 2.1624	Lux_spent
1.4830 ± 0.1643	MntSweetProducts
1.3452 ± 0.1872	MntFruits
0.6726 ± 0.0701	MntGoldProds
0.1741 ± 0.0136	RemPur
0.0752 ± 0.0065	NumPur
0.0279 ± 0.0023	Responsiveness
... 40 more ...	

Significant features are: Amount_spent, MntMeatPriducts, MntWines, Lux_spent

Section 03: Data Visualization

Please plot and visualize the answers to the below questions.

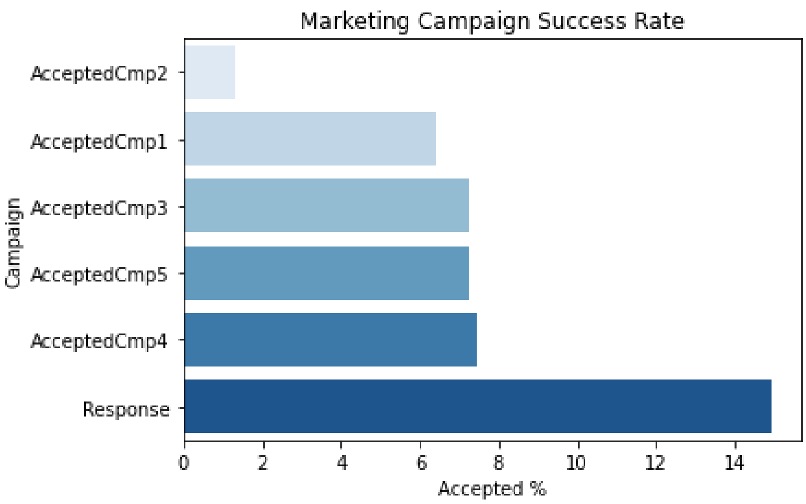
1. **Which marketing campaign is most successful?**

In [45]:

```
# Plot marketing campaign overall acceptance rates.
# Calculate success rate
campaign_success_rate= pd.DataFrame(data[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3',
'AcceptedCmp4', 'AcceptedCmp5', 'Response']].mean()*100, columns=['Percent']).reset_inde
x()

# plot
sns.barplot(x='Percent', y='index', data=campaign_success_rate.sort_values('Percent'), p
alette='Blues')
plt.xlabel('Accepted %')
plt.ylabel('Campaign')
plt.title('Marketing Campaign Success Rate', size=12)
```

Out[45]:

```
Text(0.5, 1.0, 'Marketing Campaign Success Rate')
```



Findings: The most recent campaign (column name: Response) is the most successful one.

2. What does the average customer look like for this company?

In [46]:

```
data.columns
```

Out[46]:

```
Index(['ID', 'Education', 'Marital_Status', 'Income', 'Kidhome', 'Teenhome',
       'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',
       'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
       'Response', 'Complain', 'Country', 'Age', 'Minors', 'Amount_spent',
       'Lux_spent', 'NumPur', 'RemPur', 'Responsiveness', 'Customer_since'],
      dtype='object')
```

In [47]:

```python
#age
age= round(data['Age'].mean())
print('Avg age=', age)

#income
income= round(data['Income'].mean())
print('Avg Income=', income)

#customer since
customer_since= round(data['Customer_since'].mean())
print('Customer Since=', customer_since)

#TotalAmountSpent
TotalAmountSpent= round(data['Amount_spent'].mean())
print('Avg Amount Spent=', TotalAmountSpent)

#Responsiveness
Resp= data['Responsiveness'].mean()
print('Avg Responsiveness=', Resp)

#Number of minors in household
MinorHH= data['Minors'].mean()
print('Avg no. of minors=', MinorHH)

#Educatiom
edu= data['Education'].value_counts()
print('Avg Education Qualification=', edu)

#Marital_Status
marsta= data['Marital_Status'].value_counts()
print('Marital Status=', marsta)

#Recency
rec=data['Recency'].mean()
print('Recency=', rec)
```

```
Avg age= 52
Avg Income= 52238
Customer Since= 2013
Avg Amount Spent= 606
Avg Responsiveness= 0.29776785714285714
Avg no. of minors= 0.9504464285714286
Avg Education Qualification= Graduation    1127
PhD            486
Master         370
2n Cycle       203
Basic           54
Name: Education, dtype: int64
Marital Status= Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

```
Recency= 49.109375
```

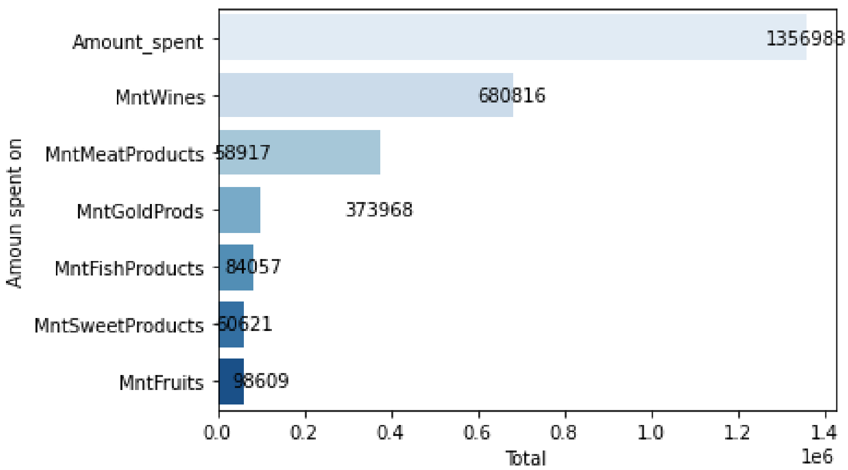
An average customer:

1. **is 52 years old**
2. **earns around 53k USD**
3. **is a custome since 2013**
4. **spent 606 USD in total**
5. **has responded to almost 0.3 campaigns**
6. **has one minor in household**
7. **is graduated**
8. **is married**
9. **made last purchase 49 days ago**

3. Which products are performing best?

In [63]:

```
mnt_cols= pd.DataFrame(data[['Amount_spent', 'MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].sum(), columns=['Total']).reset_
index()

#plot
ax=sns.barplot(x='Total', y='index', data=mnt_cols.sort_values('Total', ascending=False)
, palette='Blues')
plt.xlabel('Total')
plt.ylabel('Amoun spent on')

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



Findings: Best performing products are Wines followed by Meat Products

3.Which channels are underperforming?

In [69]:

```
data.columns
```

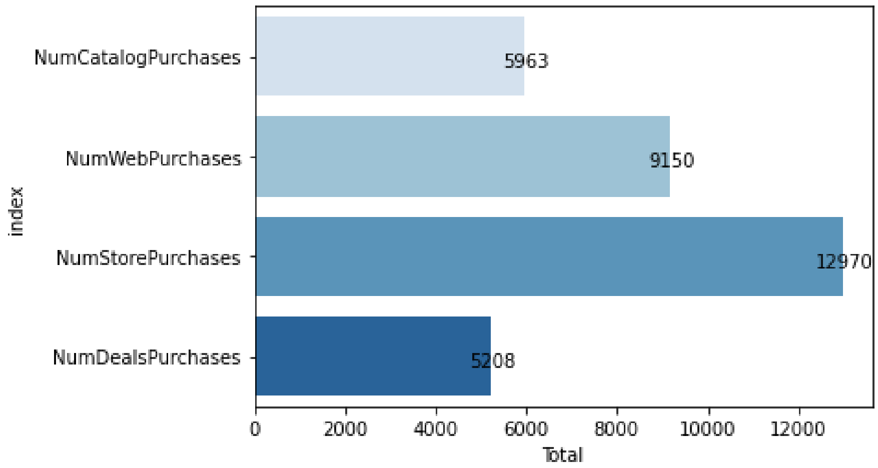
Out[69]:

```
Index(['ID', 'Education', 'Marital_Status', 'Income', 'Kidhome', 'Teenhome',
       'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',
```

```
       'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
       'Response', 'Complain', 'Country', 'Age', 'Minors', 'Amount_spent',
       'Lux_spent', 'NumPur', 'RemPur', 'Responsiveness', 'Customer_since'],
      dtype='object')
```

In [73]:

```
channels= pd.DataFrame(data[['NumCatalogPurchases', 'NumWebPurchases', 'NumStorePurchases
', 'NumDealsPurchases']].sum(), columns=['Total']).reset_index()

# plot
bx= sns.barplot(x='Total', y='index', data=channels, palette='Blues')
plt.ylabel=['Channels']
plt.xlabel=['Total']
for p,q in zip(bx.patches, channels['Total']):
  bx.text(x= q+40,
          y= p.get_y()+0.5,
          s=q,
          ha='center');
```



Findings: Catalog is the most underperforming channel followed by Deals. Store is the strongest channel.

Conclusion

- **The most successful advertising campaign was the most recent campaign (column name: Response)**
- **Advertising campaign acceptance is positively correlated with income and negatively correlated with having kids/teens**
    - **Suggested action: Create two streams of targeted advertising campaigns, one aimed at high-income individuals without kids/teens and another aimed at lower-income individuals with kids/teens**
- **The most successful products are wines and meats (i.e. the average customer spent the most on these items)**
    - **Suggested action: Focus advertising campaigns on boosting sales of the less popular items**
- **The underperforming channels are deals and catalog purchases (i.e. the average customer made the fewest purchases via these channels)**
- **The best performing channels are web and store purchases (i.e. the average customer made the most purchases via these channels)**
    - **Suggested action: Focus advertising campaigns on the more successful channels, to reach more customers**

In [ ]: