Marketing Analytics Exploratory/Statistical Analysis

Content

Section 01: Exploratory Data Analysis

- *Data Cleaning:* Identify and handle null values and outliers in the dataset through methods such as imputation or removal.
- *Feature Engineering:* Explore opportunities to create new variables from existing data that could potentially enhance predictive power or insights.

Section 02: Statistical Analysis

- *Regression Analysis:* Perform regressions to answer questions like identifying factors influencing store purchases and comparing US versus Rest of the World in terms of total purchases.
- *Hypothesis Testing:* Use appropriate statistical tests to validate or refute hypotheses, such as whether customers who spend more on gold tend to make more store purchases.

Section 03: Data Visualization

- *Campaign Success:* Visualize and compare the effectiveness of different marketing campaigns.
- *Customer Profiling:* Create visual representations of the average customer characteristics for the company.

Importing libraries and dataset

Ввод [1]:

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

df = pd.read_csv('marketing_data.csv')

df.head()
```

Out[1]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurchases	Num
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	1386	1967	Graduation	Together	$32,474.00	1	1	5/11/14	0	10	...	2	
4	5371	1989	Graduation	Single	$21,474.00	1	0	4/8/14	0	6	...	2	

5 rows × 28 columns

Ввод [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   2240 non-null   int64 
 1   Year_Birth           2240 non-null   int64 
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4    Income              2216 non-null   object
 5   Kidhome              2240 non-null   int64 
 6   Teenhome             2240 non-null   int64 
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64 
 9   MntWines             2240 non-null   int64 
 10  MntFruits            2240 non-null   int64 
 11  MntMeatProducts      2240 non-null   int64 
 12  MntFishProducts      2240 non-null   int64 
 13  MntSweetProducts     2240 non-null   int64 
 14  MntGoldProds         2240 non-null   int64 
 15  NumDealsPurchases    2240 non-null   int64 
 16  NumWebPurchases      2240 non-null   int64 
 17  NumCatalogPurchases  2240 non-null   int64 
 18  NumStorePurchases    2240 non-null   int64 
 19  NumWebVisitsMonth    2240 non-null   int64 
 20  AcceptedCmp3         2240 non-null   int64 
 21  AcceptedCmp4         2240 non-null   int64 
 22  AcceptedCmp5         2240 non-null   int64 
 23  AcceptedCmp1         2240 non-null   int64 
 24  AcceptedCmp2         2240 non-null   int64 
 25  Response             2240 non-null   int64 
 26  Complain             2240 non-null   int64 
 27  Country              2240 non-null   object
dtypes: int64(23), object(5)
memory usage: 490.1+ KB
```

Cleansing data

Ввод [3]:
```
df.columns = df.columns.str.replace(' ', '')
```

Ввод [4]:
```
df['Income'] = df['Income'].str.replace('$', '')
df['Income'] = df['Income'].str.replace(',', '').astype('float')
```

Ввод [5]:
```
df.head()
```

Out[5]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurchases	NumWe
0	1826	1970	Graduation	Divorced	84835.0	0	0	6/16/14	0	189	...	6	
1	1	1961	Graduation	Single	57091.0	0	0	6/15/14	0	464	...	7	
2	10476	1958	Graduation	Married	67267.0	0	1	5/13/14	0	134	...	5	
3	1386	1967	Graduation	Together	32474.0	1	1	5/11/14	0	10	...	2	
4	5371	1989	Graduation	Single	21474.0	1	0	4/8/14	0	6	...	2	

5 rows × 28 columns

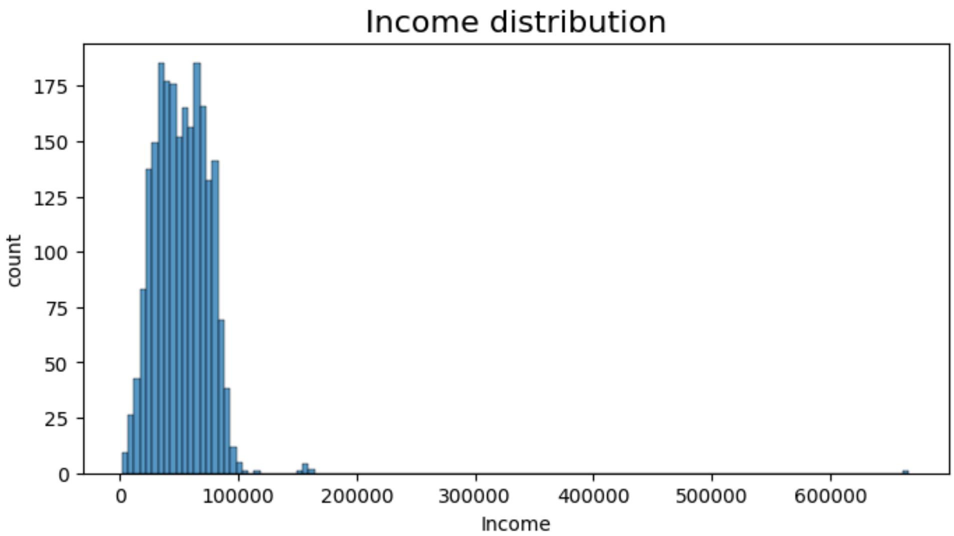
Section 01: Exploratory Data Analysis

- Redussing null values and replasing the with median values
- Transform Dt_Customer to datetime
- Designing new columns for data classification

Ввод [6]:
```
df.isnull().sum().sort_values(ascending = False)
```

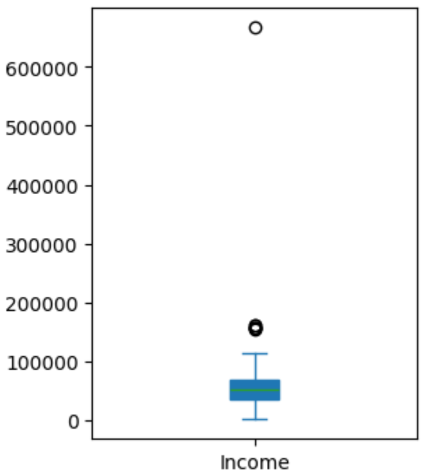
Out[6]:
```
Income                24
ID                     0
NumDealsPurchases      0
Complain               0
Response               0
AcceptedCmp2           0
AcceptedCmp1           0
AcceptedCmp5           0
AcceptedCmp4           0
AcceptedCmp3           0
NumWebVisitsMonth      0
NumStorePurchases      0
NumCatalogPurchases    0
NumWebPurchases        0
MntGoldProds           0
Year_Birth             0
MntSweetProducts       0
MntFishProducts        0
MntMeatProducts        0
MntFruits              0
MntWines               0
Recency                0
Dt_Customer            0
Teenhome               0
Kidhome                0
Marital_Status         0
Education              0
Country                0
dtype: int64
```

Ввод [7]:
```
plt.figure(figsize = (8, 4))
sns.histplot(df['Income'], kde = False)
plt.title('Income distribution', size = 16)
plt.ylabel('count');
```



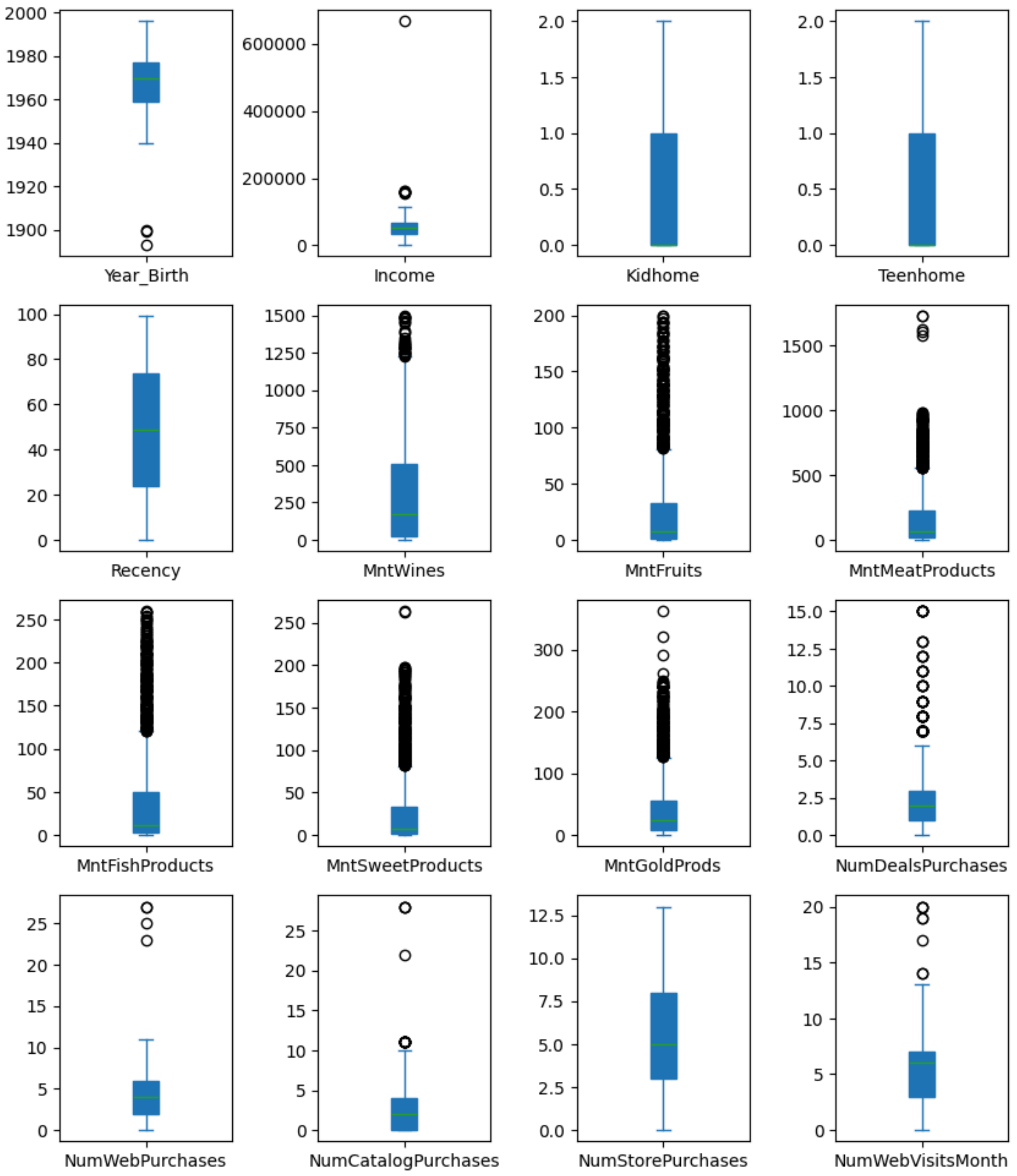
Ввод [8]:
```
df['Income'].plot(kind = 'box', figsize = (3,4), patch_artist = True)
```

Out[8]: <Axes: >



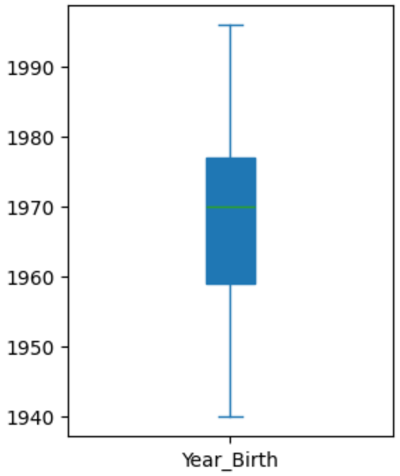
Ввод [9]:
```
df['Income'] = df['Income'].fillna(df['Income'].median())
```

Ввод [10]:
```
df_to_plot = df.drop(columns = ['ID', 'AcceptedCmp2', 'AcceptedCmp1', 'AcceptedCmp5', 'AcceptedCmp4', 'AcceptedCmp3',
                                'Response', 'Complain']).select_dtypes(include = np.number)
df_to_plot.plot(subplots = True, layout = (4, 4), kind = 'box', figsize = (10,12), patch_artist = True)
plt.subplots_adjust(wspace = 0.5)
```



Ввод [11]:
```
df = df[df['Year_Birth'] > 1900].reset_index(drop = True)
plt.figure(figsize = (3,4))
df['Year_Birth'].plot(kind = 'box', patch_artist = True)
```

Out[11]:
```
<Axes: >
```



Ввод [12]:
```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])
```

```
C:\Users\kenny\AppData\Local\Temp\ipykernel_9088\177162232.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])
```

Ввод [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2237 entries, 0 to 2236
Data columns (total 28 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2237 non-null   int64
 1   Year_Birth           2237 non-null   int64
 2   Education            2237 non-null   object
 3   Marital_Status       2237 non-null   object
 4   Income               2237 non-null   float64
 5   Kidhome              2237 non-null   int64
 6   Teenhome             2237 non-null   int64
 7   Dt_Customer          2237 non-null   datetime64[ns]
 8   Recency              2237 non-null   int64
 9   MntWines             2237 non-null   int64
 10  MntFruits            2237 non-null   int64
 11  MntMeatProducts      2237 non-null   int64
 12  MntFishProducts      2237 non-null   int64
 13  MntSweetProducts     2237 non-null   int64
 14  MntGoldProds         2237 non-null   int64
 15  NumDealsPurchases    2237 non-null   int64
 16  NumWebPurchases      2237 non-null   int64
 17  NumCatalogPurchases  2237 non-null   int64
 18  NumStorePurchases    2237 non-null   int64
 19  NumWebVisitsMonth    2237 non-null   int64
 20  AcceptedCmp3         2237 non-null   int64
 21  AcceptedCmp4         2237 non-null   int64
 22  AcceptedCmp5         2237 non-null   int64
 23  AcceptedCmp1         2237 non-null   int64
 24  AcceptedCmp2         2237 non-null   int64
 25  Response             2237 non-null   int64
 26  Complain             2237 non-null   int64
 27  Country              2237 non-null   object
dtypes: datetime64[ns](1), float64(1), int64(23), object(3)
memory usage: 489.5+ KB
```

Ввод [14]:
```
list(df.columns)
```

Out[14]:
```
['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']
```

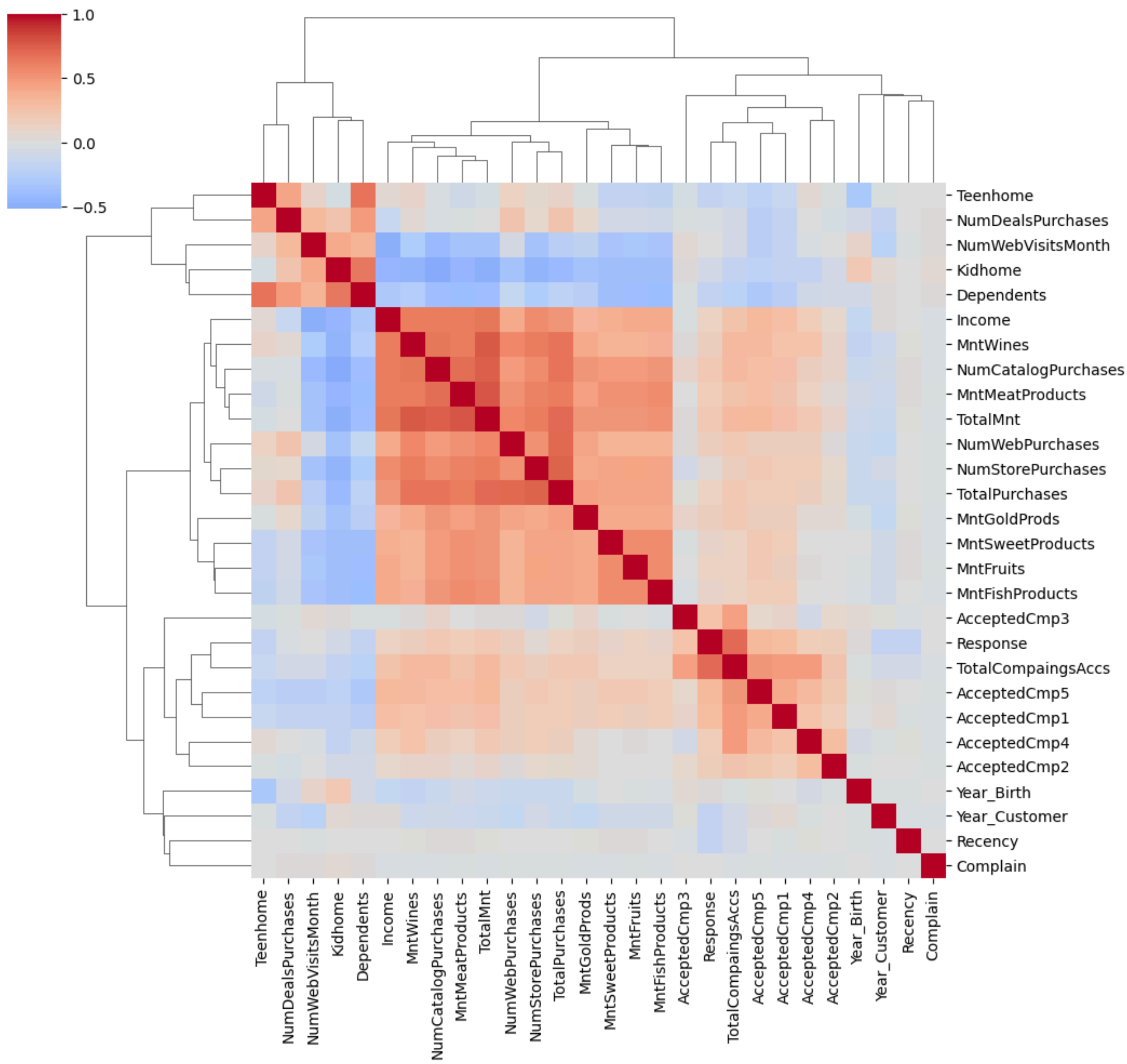
Ввод [15]:
```
df['Dependents'] = df['Kidhome'] + df['Teenhome']
df['Year_Customer'] = pd.DatetimeIndex(df['Dt_Customer']).year

mnt_cols = [col for col in df.columns if 'Mnt' in col]
df['TotalMnt'] = df[mnt_cols].sum(axis = 1)

purchase_cols = [col for col in df.columns if 'Purchases' in col]
df['TotalPurchases'] = df[purchase_cols].sum(axis = 1)

comp_cols = [col for col in df.columns if 'Cmp' in col] + ['Response']
df['TotalCompaingsAccs'] = df[comp_cols].sum(axis = 1)

df[['ID', 'Dependents', 'Year_Customer', 'TotalMnt', 'TotalPurchases', 'TotalCompaingsAccs']].head()
```

Out[15]:

	ID	Dependents	Year_Customer	TotalMnt	TotalPurchases	TotalCompaingsAccs
0	1826	0	2014	1190	15	1
1	1	0	2014	577	18	2
2	10476	1	2014	251	11	0
3	1386	2	2014	11	4	0
4	5371	1	2014	91	8	2

Ввод [16]:
```
corr = df.drop(columns = 'ID').select_dtypes(include = np.number).corr(method = 'kendall')
sns.clustermap(corr, cbar_pos = (-0.05, 0.8, 0.05, 0.18),cmap = 'coolwarm', center = 0)
```

Out[16]: <seaborn.matrix.ClusterGrid at 0x208dd011070>

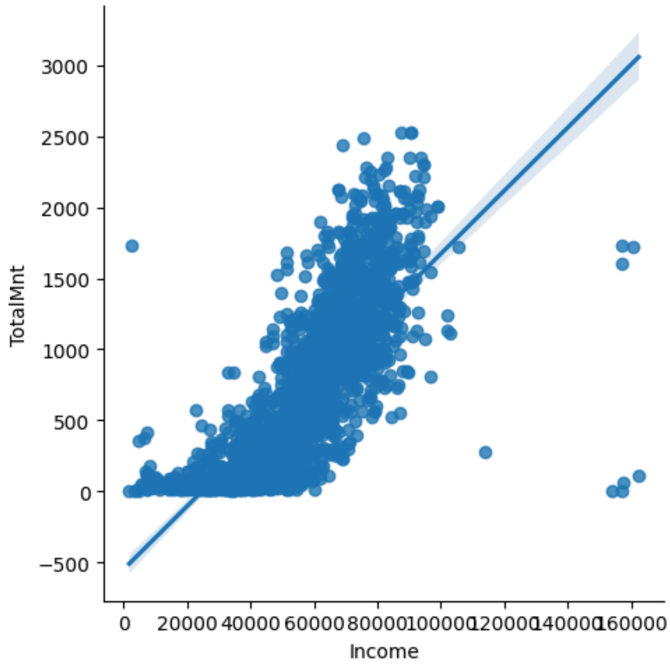


My observations:

- *High Income Cluster:*
  - Features related to *amount spent and number of purchases* are *positively correlated* with *'Income'*. This indicates that higher income customers tend to spend more and make more purchases, both in terms of total amount and across different purchase channels (store, web, catalog).
- *Have Kids & Teens Cluster:*
  - Amount spent and number of purchases are negatively correlated with 'Dependents' (specifically children). This suggests that households with more children tend to spend less and make fewer purchases. On the other hand, purchasing deals is positively correlated with 'Dependents', implying that households with kids and/or teens are more likely to take advantage of promotional deals.
- *Advertising Campaigns Cluster:*
  - Acceptance of advertising campaigns ('AcceptedCmp', 'Response') shows strong positive correlation with each other. There's also a weak positive correlation with the High Income cluster, indicating that advertising campaigns may resonate more with higher income customers. Conversely, there's a weak negative correlation with the Have Kids & Teens cluster, suggesting that these campaigns might have less impact on households with children.
- ***Anomalies:***
  - The number of website visits ('NumWebVisitsMonth') does not correlate with an increased number of web purchases ('NumWebPurchases'). Instead, it shows a positive correlation with the number of deals purchased ('NumDealsPurchases'). This anomaly suggests that while website visits are frequent, they may not directly translate into increased purchases unless promotional deals are involved.

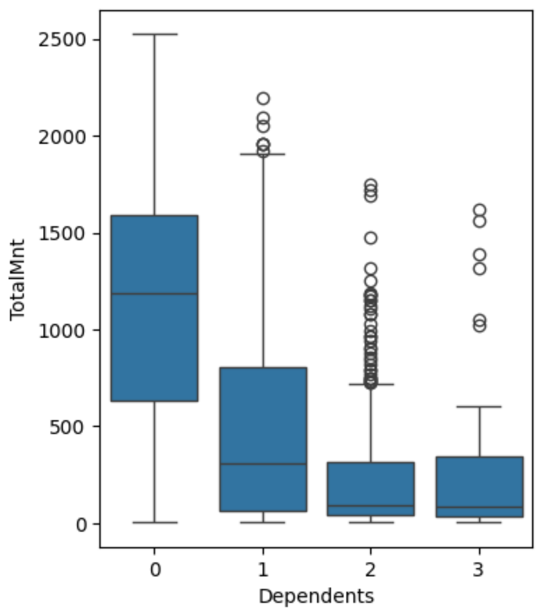
Ввод [17]:
```
sns.lmplot(x = 'Income', y = 'TotalMnt', data = df[df['Income'] < 200000])
```

Out[17]: <seaborn.axisgrid.FacetGrid at 0x208dd011a90>



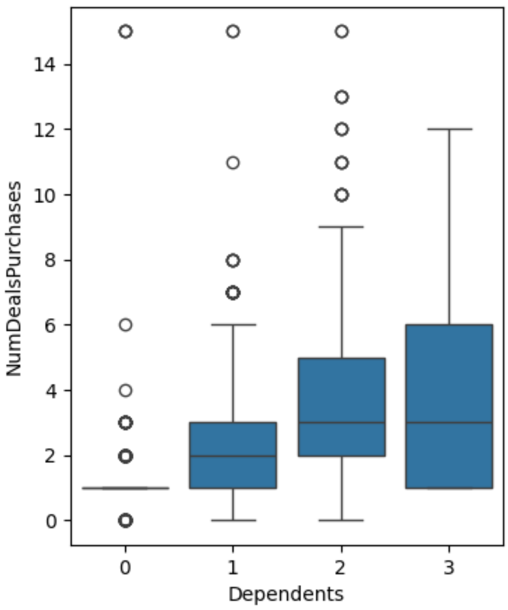
Ввод [18]:
```
plt.figure(figsize = (4,5))
sns.boxplot(x = 'Dependents', y = 'TotalMnt', data = df)
```

Out[18]: <Axes: xlabel='Dependents', ylabel='TotalMnt'>



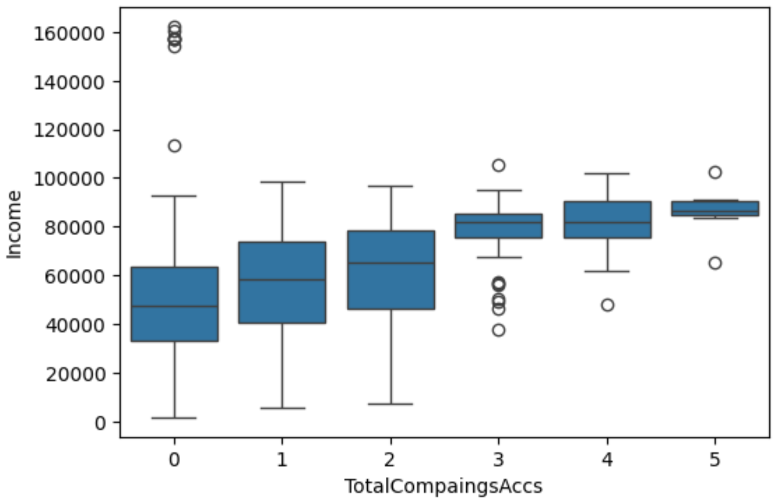
Ввод [19]:
```
plt.figure(figsize = (4,5))
sns.boxplot(x = 'Dependents', y = 'NumDealsPurchases', data = df)
```

Out[19]: <Axes: xlabel='Dependents', ylabel='NumDealsPurchases'>



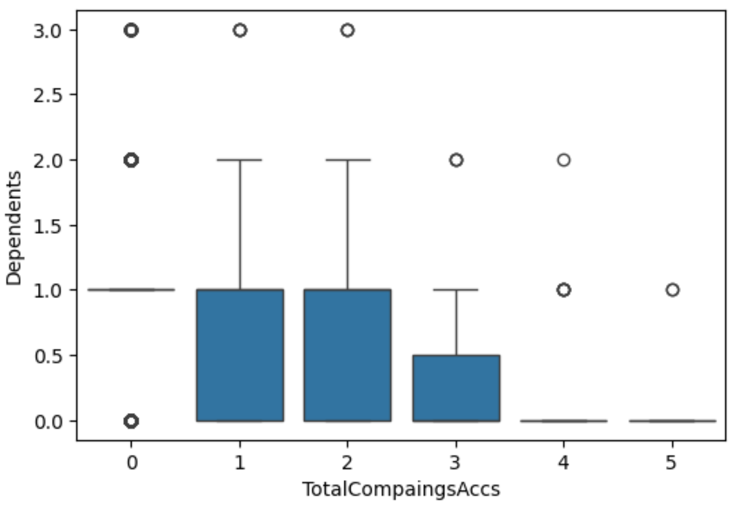
Ввод [20]:
```
plt.figure(figsize = (6,4))
sns.boxplot(x = 'TotalCompaingsAccs', y = 'Income', data = df[df['Income'] < 200000])
```

Out[20]: <Axes: xlabel='TotalCompaingsAccs', ylabel='Income'>



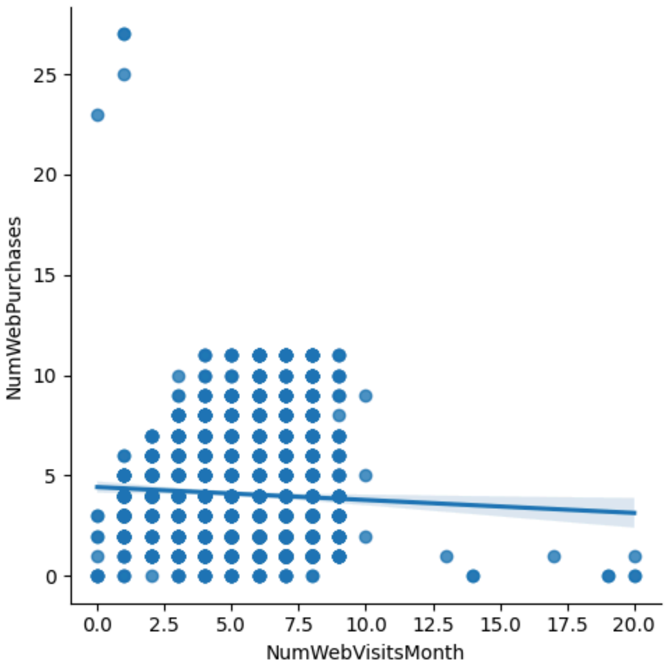
Ввод [21]:
```
plt.figure(figsize = (6,4))
sns.boxplot(x = 'TotalCompaingsAccs', y = 'Dependents', data = df)
```

Out[21]: <Axes: xlabel='TotalCompaingsAccs', ylabel='Dependents'>



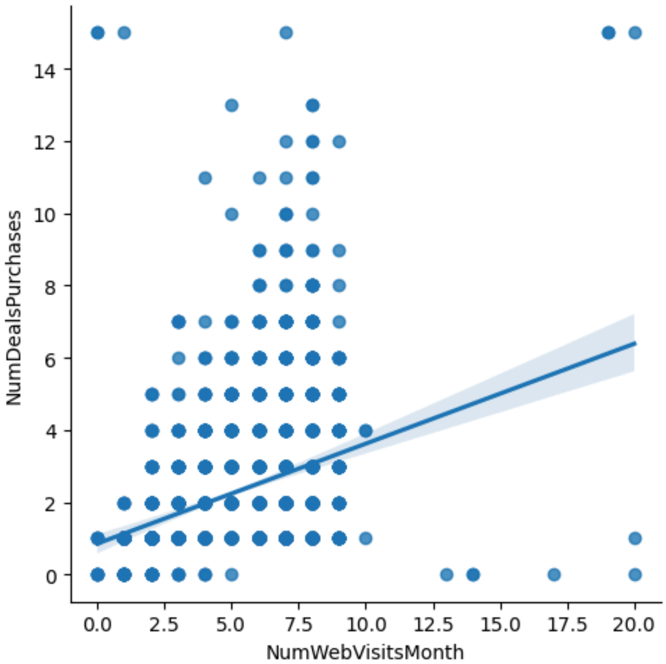
Ввод [22]:
```
sns.lmplot(x = 'NumWebVisitsMonth', y = 'NumWebPurchases', data = df)
```

Out[22]: <seaborn.axisgrid.FacetGrid at 0x208dcedc0d0>



Ввод [23]:
```
sns.lmplot(x = 'NumWebVisitsMonth', y = 'NumDealsPurchases', data = df)
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x208dd4c3d30>



Section 02: Statistical Analysis

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

Ввод [24]:
```
plt.figure(figsize = (6,4))

sns.histplot(df['NumStorePurchases'], kde = False, bins = 12)
plt.title('NumStorePurchases distribution', size = 16)
plt.ylabel('count')
```

Out[24]: Text(0, 0.5, 'count')

Ввод [25]:
```
df.drop(columns = ['ID', 'Dt_Customer'], inplace = True)
```

Ввод [26]:
```
#conda update scikit-learn
```

Ввод [27]:
```
from sklearn.preprocessing import OneHotEncoder

cat = df.select_dtypes(exclude = np.number)

print('Num of unique values per categorical features: \n', cat.nunique())

enc = OneHotEncoder()
cat_encoded = enc.fit_transform(cat).toarray()

feature_names = enc.get_feature_names_out(cat.columns)
cat_encoded = pd.DataFrame(cat_encoded, columns = feature_names)

num = df.drop(columns = cat.columns)
df2 = pd.concat([cat_encoded, num], axis = 1)
df2.head()
```

```
Num of unique values per categorical features:
 Education         5
Marital_Status    8
Country           8
dtype: int64
```

Out[27]:

	Education_2n Cycle	Education_Basic	Education_Graduation	Education_Master	Education_PhD	Marital_Status_Absurd	Marital_Status_Alone	Marital_S
0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	

5 rows × 49 columns

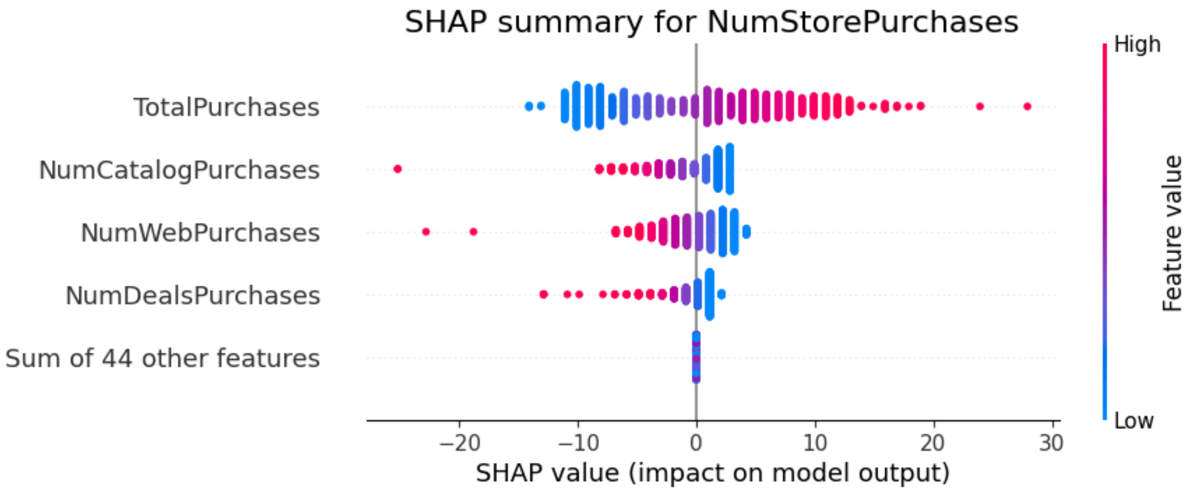
Ввод [28]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X = df2.drop(columns = 'NumStorePurchases')
y = df2['NumStorePurchases']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
                                                    random_state = 1)
model = LinearRegression()
model.fit(X_train, y_train)

preds = model.predict(X_test)

print('Linear Regression Model RMSE:', np.sqrt(mean_squared_error(y_test, preds)))
print('Median value of target variable:', y.median())
```

```
Linear Regression Model RMSE: 3.436436081622267e-14
Median value of target variable: 5.0
```

Ввод [29]:
```
# import eli5
# from eli5.sklearn import PermutationImportance

# perm = PermutationImportance(model, random_state = 1).fit(X_test, y_test)

# eli5.show_weight(perm, feature_names = X_test.columns.tolist(), top = 5)
```

Ввод [30]:
```
#pip install eli5
#pip install --upgrade scikit-learn
#pip install --upgrade eli5
#pip uninstall scikit-learn
#pip install scikit-learn==0.23.2
```

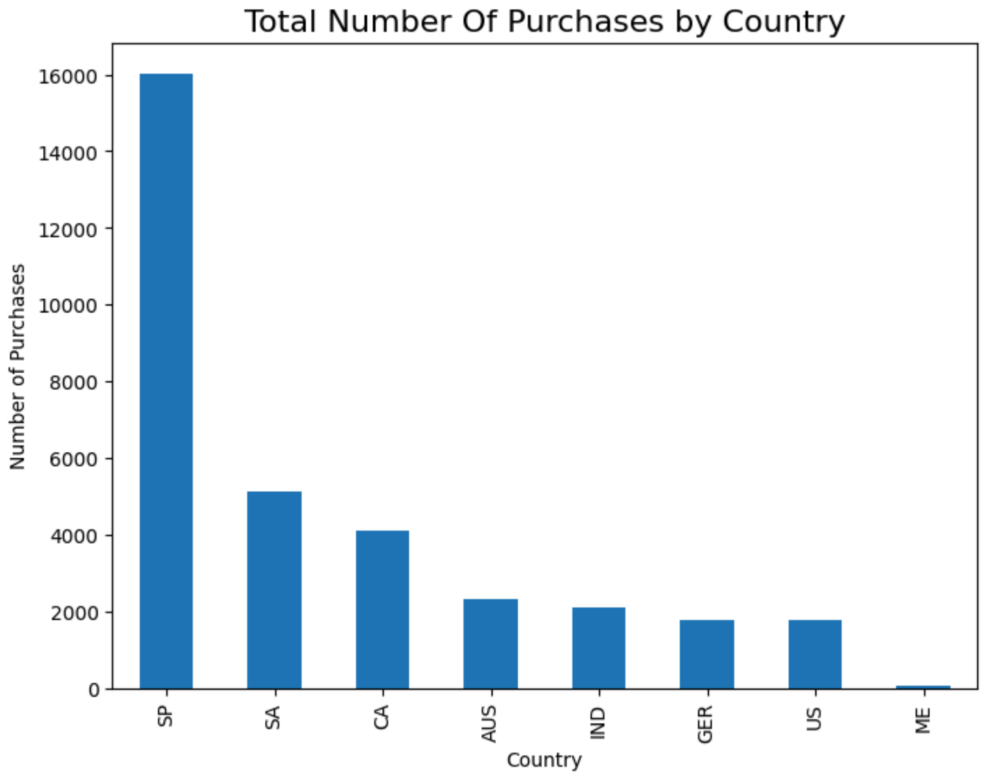
Ввод [31]:
```
import shap

ex = shap.Explainer(model, X_train)
shap_values = ex(X_test)

plt.title('SHAP summary for NumStorePurchases', size=16)
shap.plots.beeswarm(shap_values, max_display=5);
```



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

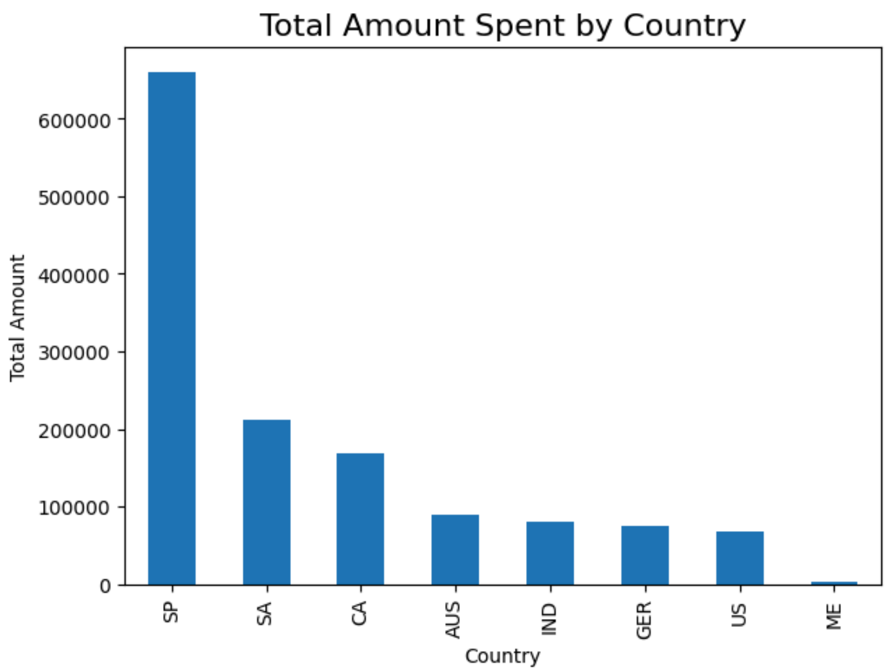
Ввод [32]:
```
plt.figure(figsize = (8,6))
df.groupby('Country')['TotalPurchases'].sum().sort_values(ascending = False).plot(kind = 'bar')

plt.title('Total Number Of Purchases by Country', size = 16)
plt.ylabel('Number of Purchases')
```

Out[32]: Text(0, 0.5, 'Number of Purchases')



Ввод [33]:
```
plt.figure(figsize = (7,5))
df.groupby('Country')['TotalMnt'].sum().sort_values(ascending = False).plot(kind = 'bar')

plt.title('Total Amount Spent by Country', size = 16)
plt.ylabel('Total Amount')
```

Out[33]: Text(0, 0.5, 'Total Amount')

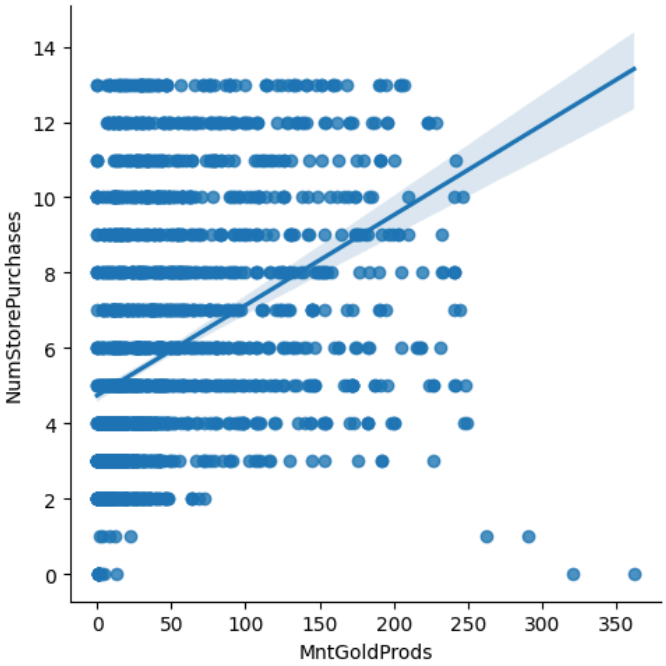


Is there any correlation between spending above average on gold and purchasing more in a store? is it statistically significant?

Ввод [34]:
```
plt.figure(figsize = (6,4))
sns.lmplot(x = 'MntGoldProds', y = 'NumStorePurchases', data = df)
```

Out[34]: <seaborn.axisgrid.FacetGrid at 0x208e2ec7880>

<Figure size 600x400 with 0 Axes>



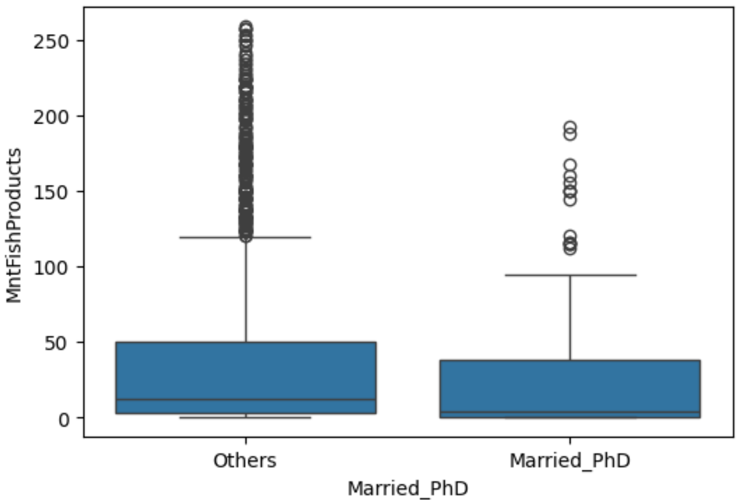
Ввод [35]:
```
from scipy.stats import kendalltau

kendall_corr = kendalltau(x=df['MntGoldProds'], y=df['NumStorePurchases'])

print('Kendall correlation (tau):', kendall_corr.correlation)
print('Kendall p-value', kendall_corr.pvalue)
```

```
Kendall correlation (tau): 0.3927160395725131
Kendall p-value 3.5588181790543497e-152
```

Do "Married PhD candidates" have a significant relation with amount spent on fish?

Ввод [36]:
```
df2['Married_PhD'] = df2['Marital_Status_Married'] + df2['Education_PhD']
df2['Married_PhD'] = df2['Married_PhD'].replace({2:'Married_PhD', 1:'Others', 0:'Others'})

plt.figure(figsize = (6,4))
sns.boxplot(x = 'Married_PhD', y = 'MntFishProducts', data = df2)
```

Out[36]: <Axes: xlabel='Married_PhD', ylabel='MntFishProducts'>

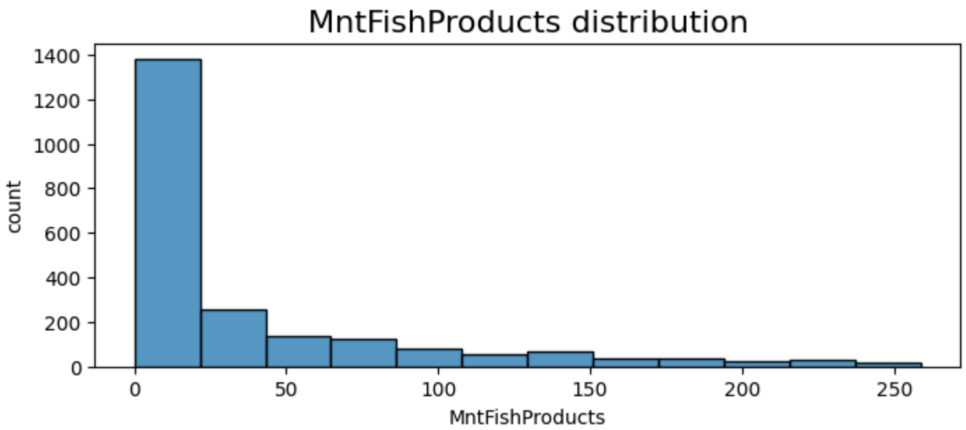


Ввод [37]:
```
from scipy.stats import ttest_ind

pval = ttest_ind(df2[df2['Married_PhD'] == 'Married_PhD']['MntFishProducts'], df2[df2['Married_PhD'] == 'Others']['Mnt

print('t-test p-values', round(pval, 3))
```

```
t-test p-values 0.005
```

Ввод [38]:
```
plt.figure(figsize = (8,3))

sns.histplot(df['MntFishProducts'], kde = False, bins = 12)
plt.title('MntFishProducts distribution', size = 16)
plt.ylabel('count')
```

Out[38]:
```
Text(0, 0.5, 'count')
```

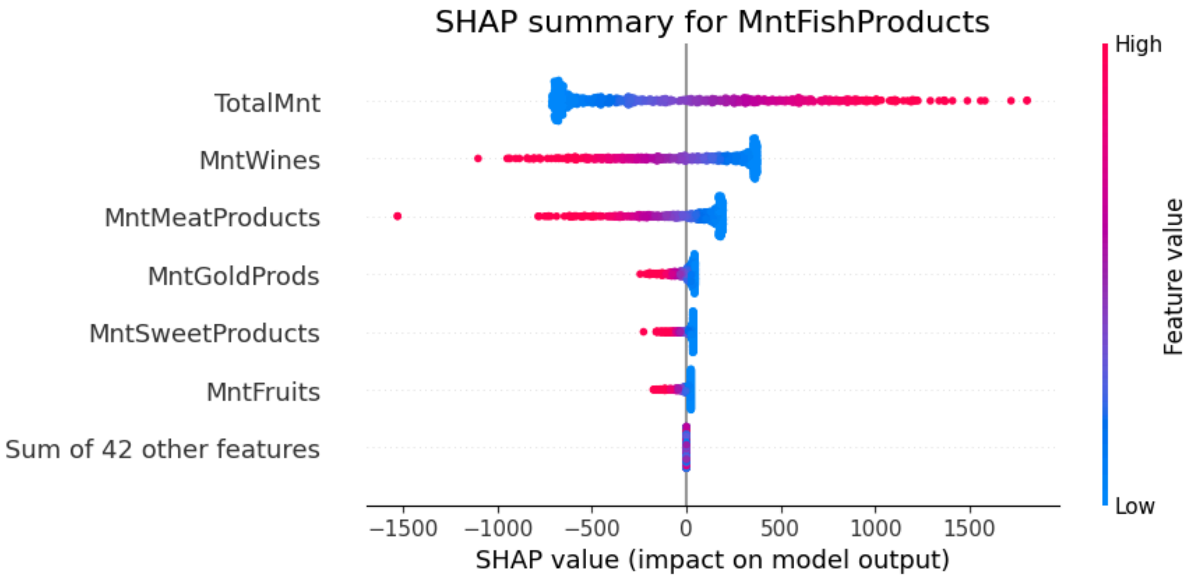


Ввод [39]:
```
df2.drop(columns='Married_PhD', inplace=True)
```

Ввод [40]:
```
X = df2.drop(columns='MntFishProducts')
y = df2['MntFishProducts']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)

model = LinearRegression()
model.fit(X_train, y_train)

preds = model.predict(X_test)

print('Linear regression model RMSE:', np.sqrt(mean_squared_error(y_test, preds)))
print('Median value of target variable:', y.median())
```

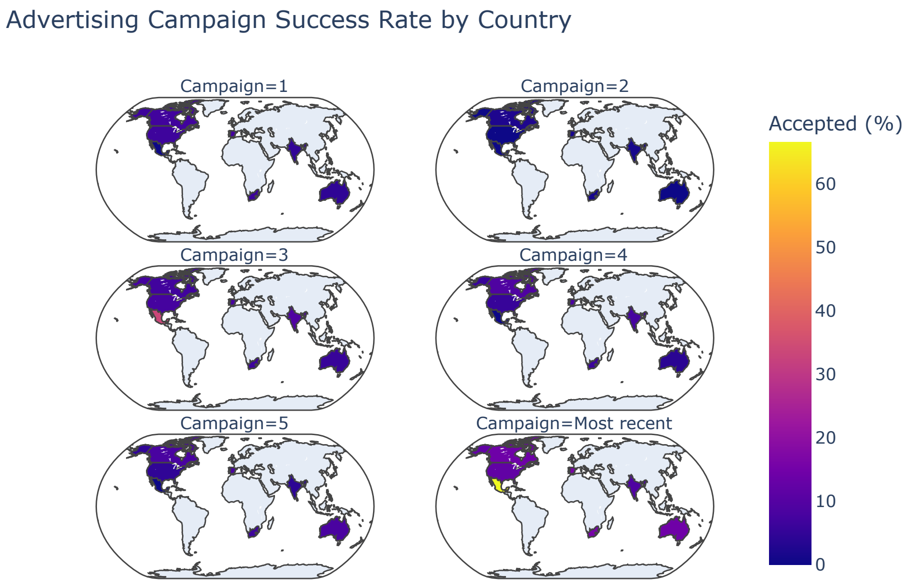
```
Linear regression model RMSE: 6.597346993307554e-13
Median value of target variable: 12.0
```

Ввод [41]:
```
ex = shap.Explainer(model, X_train)

shap_value = ex(X_test)

plt.figure(figsize = (8,3))
plt.title('SHAP summary for MntFishProducts', size = 16)
shap.plots.beeswarm(shap_value, max_display = 7)
```



Is there a significant relationship between geographical regional and success of a campaign?

Ввод [42]:
```
df['Country_code'] = df['Country'].replace({'SP': 'ESP', 'CA': 'CAN',
                                            'US': 'USA', 'SA': 'ZAF', 'ME': 'MEX'})

df_cam = df[['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()

df_cam['Campaign'] = df_cam['Campaign'].replace({'AcceptedCmp1': '1', 'AcceptedCmp2': '2', 'AcceptedCmp3': '3',
                                                  'AcceptedCmp4': '4', 'AcceptedCmp5': '5', 'Response': 'Most recent'}

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 Rat
fig.show()
```



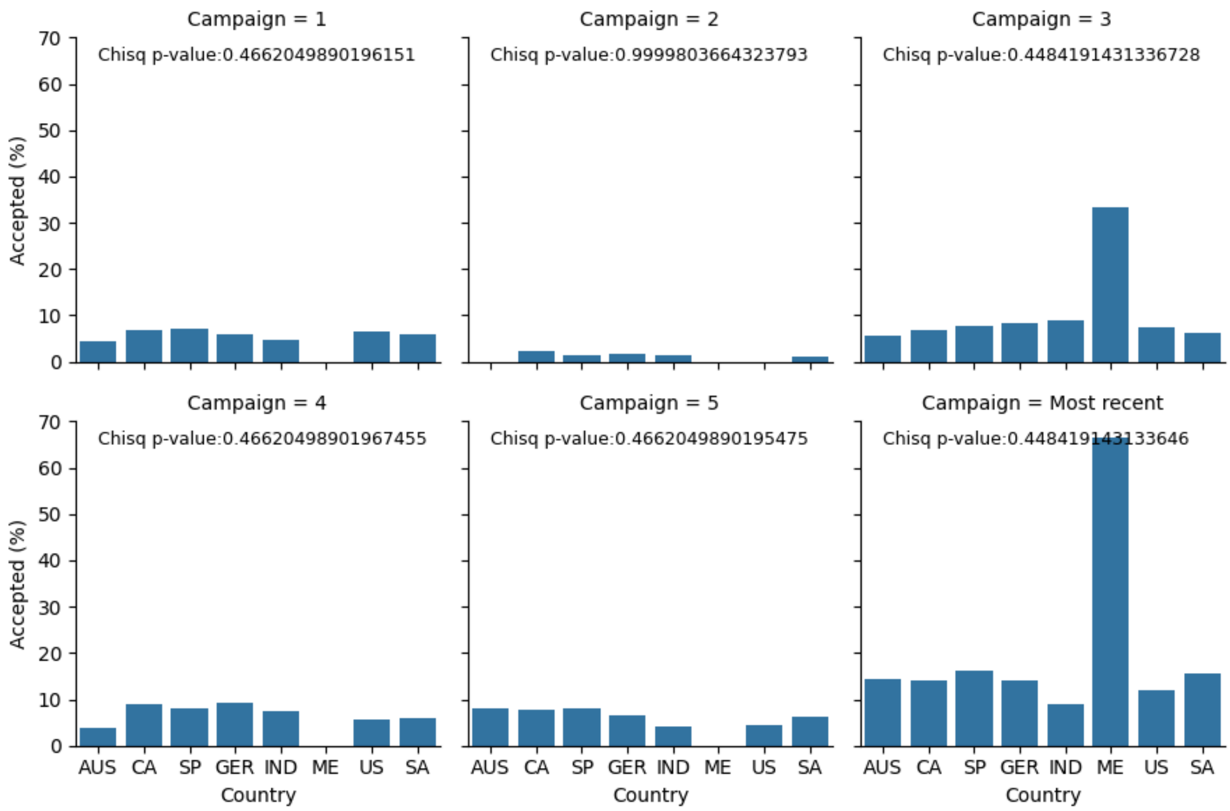
Ввод [43]:

```
import statsmodels.formula.api as smf
from scipy import stats
import statsmodels.api as sm

df_cam_wide = df[['Country', 'AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4',
                  'AcceptedCmp5', 'Response']]
stat_results = []

for col in df_cam_wide.drop(columns = 'Country').columns:
    this_data = df_cam_wide[['Country', col]]
    formula = f"{col} ~ C(Country)"

    model = smf.glm(formula=formula, data=this_data, family=sm.genmod.families.Binomial())
    results = model.fit()
    chisq = results.pearson_chi2

    pval = stats.distributions.chi2.sf(chisq, results.df_resid)
    stat_results.append(pval)

    print(results.summary())
    print(f"Chi-square p-value for {col} vs Country: {pval}\n")
    print("\nChisq p-values:", stat_results)
```

```
Time:                        14:39:06   Pearson chi2:                 2.23e+03
No. Iterations:                    20   Pseudo R-squ. (CS):           0.001563
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            -3.0845      0.387     -7.980      0.000      -3.842      -2.327
C(Country)[T.CA]      0.4534      0.457      0.992      0.321      -0.442       1.349
C(Country)[T.GER]     0.3031      0.549      0.552      0.581      -0.772       1.379
C(Country)[T.IND]     0.0888      0.547      0.162      0.871      -0.984       1.161
C(Country)[T.ME]    -18.4815   1.69e+04     -0.001      0.999   -3.31e+04    3.31e+04
C(Country)[T.SA]      0.3245      0.450      0.721      0.471      -0.558       1.207
C(Country)[T.SP]      0.5176      0.404      1.281      0.200      -0.274       1.309
C(Country)[T.US]      0.4055      0.550      0.738      0.461      -0.672       1.483
=====================================================================================
Chi-square p-value for AcceptedCmp1 vs Country: 0.4662049890196151


Chisq p-values: [0.4662049890196151]
                 Generalized Linear Model Regression Results
```

Ввод [44]:
```
countries = df[['Country', 'Country_code']].drop_duplicates().reset_index(drop = True)
df_cam2 = df_cam.merge(countries, how = 'left', on = 'Country_code')

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

for ax, pval in zip(g.axes.flat, stat_results):
    ax.text(0, 65, 'Chisq p-value:'+str(pval), fontsize = 9)
```

```
C:\Users\kenny\anaconda3\envs\notebook\lib\site-packages\seaborn\axisgrid.py:718: UserWarning:

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



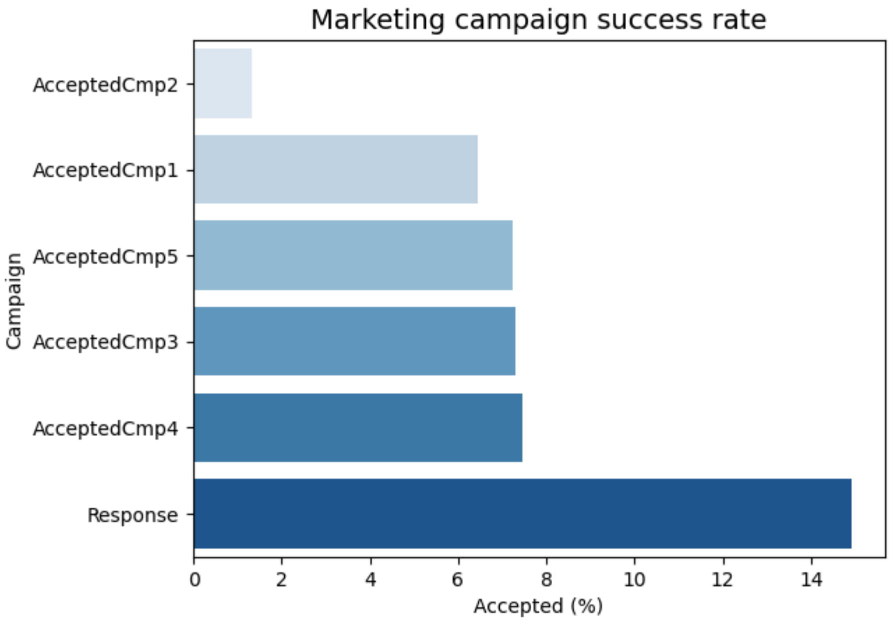
Section 03: Data Visualization

Which marketing campaign is most successful?

Ввод [45]:
```
cam_success = pd.DataFrame(df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Respon
sns.barplot(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 = 14)
```

```
C:\Users\kenny\AppData\Local\Temp\ipykernel_9088\2710237949.py:2: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `h
ue` and set `legend=False` for the same effect.
```

Out[45]:
```
Text(0.5, 1.0, 'Marketing campaign success rate')
```



What does the average customer look like for this company?

Ввод [46]:
```
binary_cols = [col for col in df.columns if 'Accepted' in col] + ['Response', 'Complain']

mnt_cols = [col for col in df.columns if 'Mnt' in col]

channel_cols = [col for col in df.columns if 'Num' in col] + ['TotalPurchases', 'TotalCompaingsAccs']
```

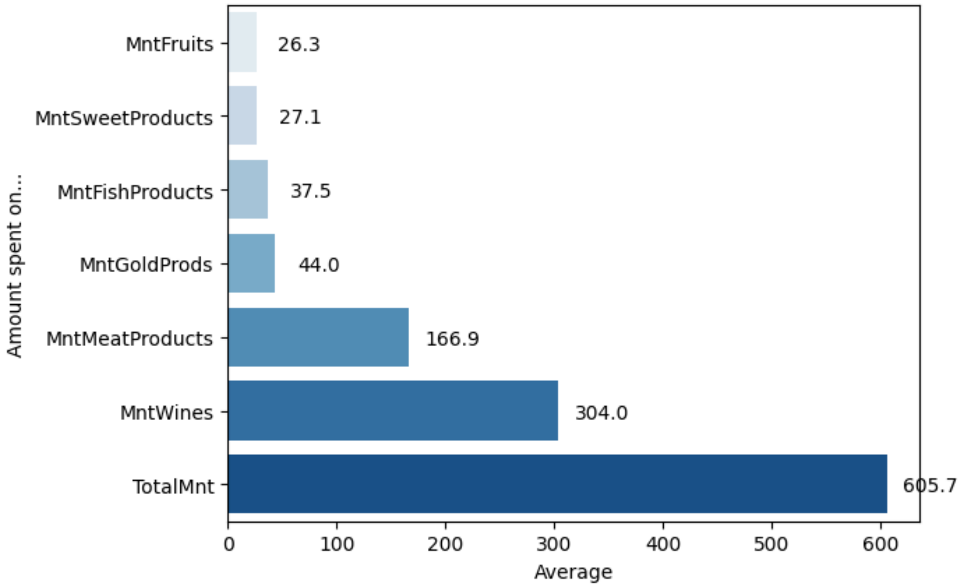
Ввод [47]:
```
num_cols = df.select_dtypes(include = np.number).columns.tolist()
num_cols = [col for col in num_cols if col not in binary_cols + mnt_cols + channel_cols]
```

Ввод [48]:
```
demographics = pd.DataFrame(round(df[num_cols].mean(), 1), columns=['Average']).reindex([
    'Year_Birth', 'Year_Customer', 'Income', 'Dependents', 'Kidhome', 'Teenhome', 'Recency'])

demographics
```

Out[48]:

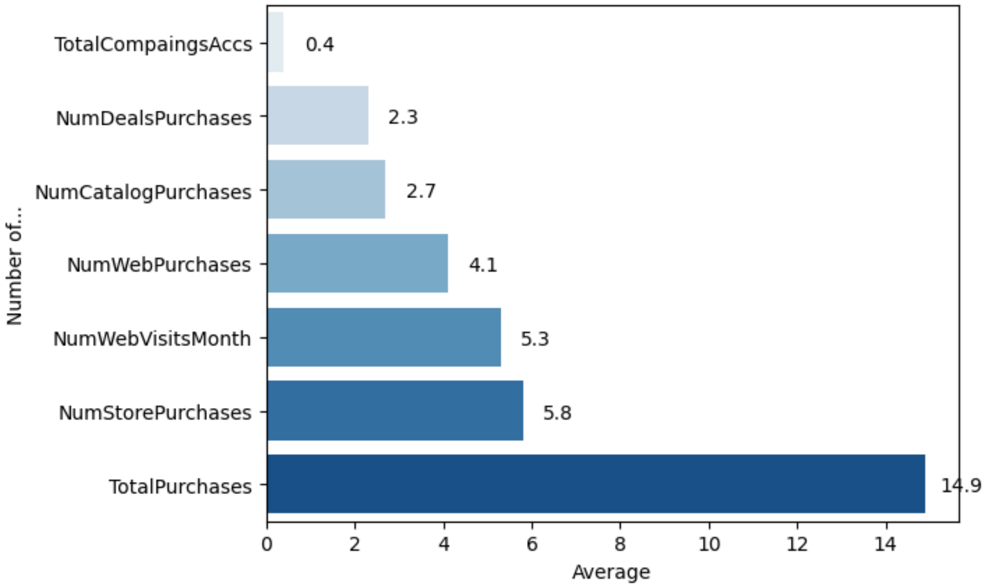
	Average
Year_Birth	1968.9
Year_Customer	2013.0
Income	52227.4
Dependents	1.0
Kidhome	0.4
Teenhome	0.5
Recency	49.1

Ввод [49]:
```
spending = pd.DataFrame(round(df[mnt_cols].mean(), 1),
                        columns = ['Average']).sort_values(by = 'Average').reset_index()

spending.rename(columns = {'index' : 'Product'}, inplace = True)

ax = sns.barplot(x = 'Average', y = 'Product', data = spending, hue = 'Product', palette = 'Blues', dodge = False, leg
plt.ylabel('Amount spent on...')

for p, q in zip(ax.patches, spending['Average']):
    ax.text(x = q + 40,
            y =p.get_y() + 0.5,
            s = q,
            ha='center')
```



Which channels are underperforming?

Ввод [50]:
```
channels = pd.DataFrame(round(df[channel_cols].mean(), 1), columns=['Average']).sort_values(by='Average').reset_index(
channels.rename(columns = {'index' : 'Product'}, inplace = True)

ax = sns.barplot(x = 'Average', y = 'Product', data = channels, hue = 'Product', dodge = False, palette = 'Blues', leg
plt.ylabel('Number of...')

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')
```



Summary Findings and Recommendations:

1. Successful Advertising Campaign:

- The most recent advertising campaign (labeled as "Response") was highly successful in Mexico, achieving an acceptance rate of over 60%.
- ***Recommendation:*** Future campaigns should emulate the successful model used in Mexico.

2. Top-Selling Products:

- Customers spend the most on wines and meats.
- ***Recommendation:*** Focus on increasing sales of less popular items to diversify revenue streams.

3. Best Performing Sales Channels:

- Web and store purchases are the most successful.
- ***Recommendation:*** Prioritize advertising efforts on these channels to maximize customer reach.

4. Underperforming Channels:

- Deals and catalog purchases have the lowest engagement.
- ***Recommendation:*** Reevaluate the effectiveness of these channels and consider reallocating resources to more successful ones.

By implementing these recomendations, the marketing department can enhance campaign effectiveness, improve customer engagement, and drive overall sales and profitability.