Customer and Platforms Analysis with Clustering

This Exploratory Data Analysis uses the Customer Personality Analysis dataset from Kaggle. It also adopts this methodology from TheCleverProgrammer.

This EDA and visualization will perform feature engineering and clustering to identify key characteristics of the company's customers and to consequently highlight c best marketing approaches and customer concerns.

Download Dataset from Kaggle

opendatasets module may be used to download the dataset directly using the Kaggle URL. This will require your Kaggle username and Kaggle Key to proceed.

In [1]:
```
import opendatasets as od
download_url = 'https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis?select=marketing_campaign.csv'
od.download(download_url, Force=True)
data_filename = './customer-personality-analysis/marketing_campaign.csv'
```

```
Skipping, found downloaded files in ".\customer-personality-analysis" (use force=True to force download)
```

Import the necessary libraries

Be sure to have pip install each of the libraries and modules before importing.

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.mixture import GaussianMixture
import datetime
from datetime import date
import plotly.graph_objects as go
from sklearn.preprocessing import StandardScaler, normalize
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv(data_filename, sep='\t')
```

Data pre-processing

Conduct data pre-processing by checking the shape of dataset, data types, missing values, outliers, and duplicate values.

In [4]:
```
print(df.columns)
print(df.info())
print(df.shape)
```

```
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', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 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   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  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
None
(2240, 29)
```

In [ ]:

Observations

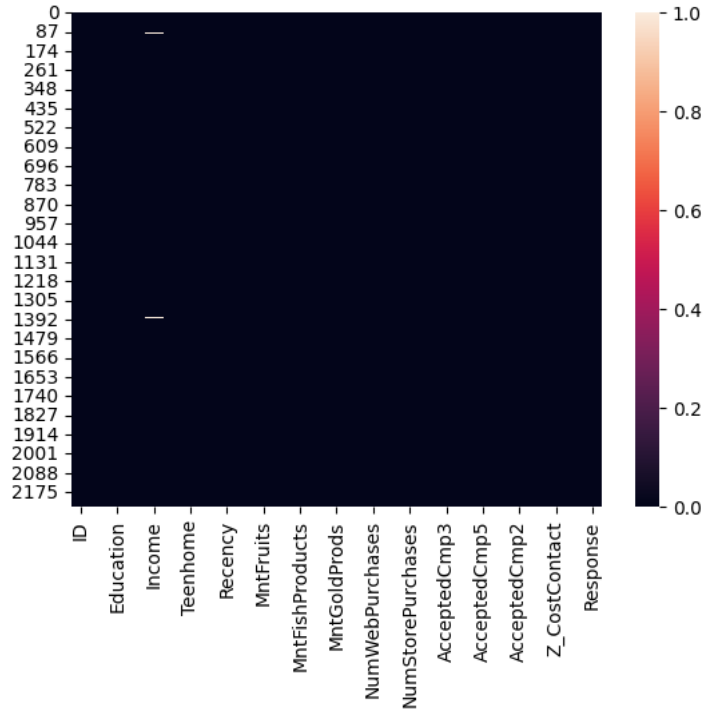
1. There are 29 columns and 2240 entries.
2. There are missing values in the 'Income' column.
3. The data types are a combination of object, float, and int.

Observations: Missing Values

Only a small proportion of 'Income' is missing, at approximately 1%.

In [5]:
```
null_inc = sns.heatmap(df.isnull(), cbar='False')
print("Proportion of 'Income' missing: " + str(df['Income'].isna().mean()))
```

```
Proportion of 'Income' missing: 0.010714285714285714
```



Observation: Outlier

There is one (1) outlier in the data with an income greater than 600,000. It may be removed.

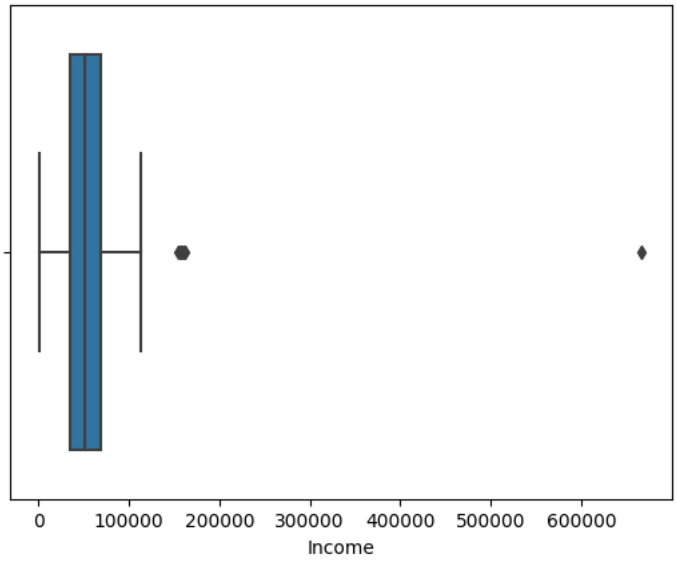
In [6]:
```
sns.boxplot(x = df['Income'])
print(df[df.Income>600000].info)
df.loc[2233:2240, :]
```

```
<bound method DataFrame.info of         ID  Year_Birth   Education Marital_Status    Income  Kidhome  \
2233  9432        1977  Graduation       Together  666666.0        1

      Teenhome Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  \
2233         0  02-06-2013       23         9  ...                  6

      AcceptedCmp3  AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  \
2233             0             0             0             0             0

      Complain  Z_CostContact  Z_Revenue  Response
2233         0              3         11         0

[1 rows x 29 columns]>
```

Out[6]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	AcceptedCmp3	Accep
2233	9432	1977	Graduation	Together	666666.0	1	0	02-06-2013	23	9	...	6	0	
2234	8372	1974	Graduation	Married	34421.0	1	0	01-07-2013	81	3	...	7	0	
2235	10870	1967	Graduation	Married	61223.0	0	1	13-06-2013	46	709	...	5	0	
2236	4001	1946	PhD	Together	64014.0	2	1	10-06-2014	56	406	...	7	0	
2237	7270	1981	Graduation	Divorced	56981.0	0	0	25-01-2014	91	908	...	6	0	
2238	8235	1956	Master	Together	69245.0	0	1	24-01-2014	8	428	...	3	0	
2239	9405	1954	PhD	Married	52869.0	1	1	15-10-2012	40	84	...	7	0	

7 rows × 29 columns



In [7]:
```
df = df.drop([2233], axis=0)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2239 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2239 non-null   int64
 1   Year_Birth           2239 non-null   int64
 2   Education            2239 non-null   object
 3   Marital_Status       2239 non-null   object
 4   Income               2215 non-null   float64
 5   Kidhome              2239 non-null   int64
 6   Teenhome             2239 non-null   int64
 7   Dt_Customer          2239 non-null   object
 8   Recency              2239 non-null   int64
 9   MntWines             2239 non-null   int64
 10  MntFruits            2239 non-null   int64
 11  MntMeatProducts      2239 non-null   int64
 12  MntFishProducts      2239 non-null   int64
 13  MntSweetProducts     2239 non-null   int64
 14  MntGoldProds         2239 non-null   int64
 15  NumDealsPurchases    2239 non-null   int64
 16  NumWebPurchases      2239 non-null   int64
 17  NumCatalogPurchases  2239 non-null   int64
 18  NumStorePurchases    2239 non-null   int64
 19  NumWebVisitsMonth    2239 non-null   int64
 20  AcceptedCmp3         2239 non-null   int64
 21  AcceptedCmp4         2239 non-null   int64
 22  AcceptedCmp5         2239 non-null   int64
 23  AcceptedCmp1         2239 non-null   int64
 24  AcceptedCmp2         2239 non-null   int64
 25  Complain             2239 non-null   int64
 26  Z_CostContact        2239 non-null   int64
 27  Z_Revenue            2239 non-null   int64
 28  Response             2239 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 524.8+ KB
```

Observation: Object data type for Dates

The 'Dt_Customer' field has dates in the format dd-mm-yyyy but it is in object data type.

In [8]:
```
df['Dt_Customer'].head(10)
```

Out[8]:
```
0    04-09-2012
1    08-03-2014
2    21-08-2013
3    10-02-2014
4    19-01-2014
5    09-09-2013
6    13-11-2012
7    08-05-2013
8    06-06-2013
9    13-03-2014
Name: Dt_Customer, dtype: object
```

Feature Engineering

We will now create new features and modify existing features that my analysis will draw from. To be precise, I will:

- Create an Age feature using 'Year_Birth'
- Create a Bill feature for a total amount spent for each food category
- Create a Seniority feature using Dt_Customer
- Replace categories in 'Marital_Status', 'Education'
- Rename fields in Mntspent, and Campaign acceptance
- Create Children, Is_Parent, and Household features

In [9]:
```
# Age feature
df['Age']= 2014-df['Year_Birth']

# Bill feature
MntSpent= list(df[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntSweetProducts', 'MntGoldProds']])
df['Bill']= df[MntSpent].sum(axis=1)

# Seniority feature
date_final= date(2014,10, 4)
df['Seniority']= pd.to_datetime(df['Dt_Customer'], dayfirst=True,format='%d-%m-%Y')
df['Seniority']= pd.to_numeric(df['Seniority'].dt.date.apply(lambda x: (date_final - x)).dt.days, downcast='integer')/30

# Replace Marital_Status categories for standardization and clarity
df['Marital_Status']= df['Marital_Status'].replace({'Married':'In couple','Together':'In couple','Absurd':'Alone','Widow':'Alone','YOLO':'Alon

# Replace Education categories for standardization and clarity
df['Education']= df['Education'].replace({'Basic':'Undergraduate','2n Cycle':'Undergraduate','Graduation':'Postgraduate','Master':'Postgraduat

# Rename Mntspent fields for clarity
df= df.rename(columns={'NumWebPurchases': 'Web','NumCatalogPurchases':'Catalog','NumStorePurchases':'Store'})

# Out-of-store purchases
df['Out-Store'] = df['Web'] + df['Catalog']

#Rename campaign fields for clarity
df = df.rename(columns={'AcceptedCmp1': 'Campaign_1', 'AcceptedCmp2': 'Campaign_2', 'AcceptedCmp3': 'Campaign_3', 'AcceptedCmp4': 'Campaign_4'

# Create Children, Is_Parent, and Household features
df['Children'] = df['Kidhome'] + df['Teenhome']
df['Is_Parent'] = np.where(df.Children> 0, 1, 0)
df['Household'] = df['Marital_Status'].replace({'Alone': 1, 'In couple':2})+ df['Children']

df = df[['Age', 'Bill', 'Income','Seniority', 'Marital_Status', 'Education', 'Out-Store', 'Store', 'Children', 'Is_Parent', 'Household', 'Camp
df.head()
```

Out[9]:

	Age	Bill	Income	Seniority	Marital_Status	Education	Out-Store	Store	Children	Is_Parent	Household	Campaign_1	Campaign_2	Campaign_3	Campaign_4
0	57	1445	58138.0	25.333333	Alone	Postgraduate	18	4	0	0	1	0	0	0	0
1	60	25	46344.0	7.000000	Alone	Postgraduate	2	2	2	1	3	0	0	0	0
2	49	665	71613.0	13.633333	In couple	Postgraduate	10	10	0	0	2	0	0	0	0
3	30	43	26646.0	7.866667	In couple	Postgraduate	2	4	1	1	3	0	0	0	0
4	33	376	58293.0	8.600000	In couple	Postgraduate	8	6	1	1	3	0	0	0	0

Visualizing the customer demographic

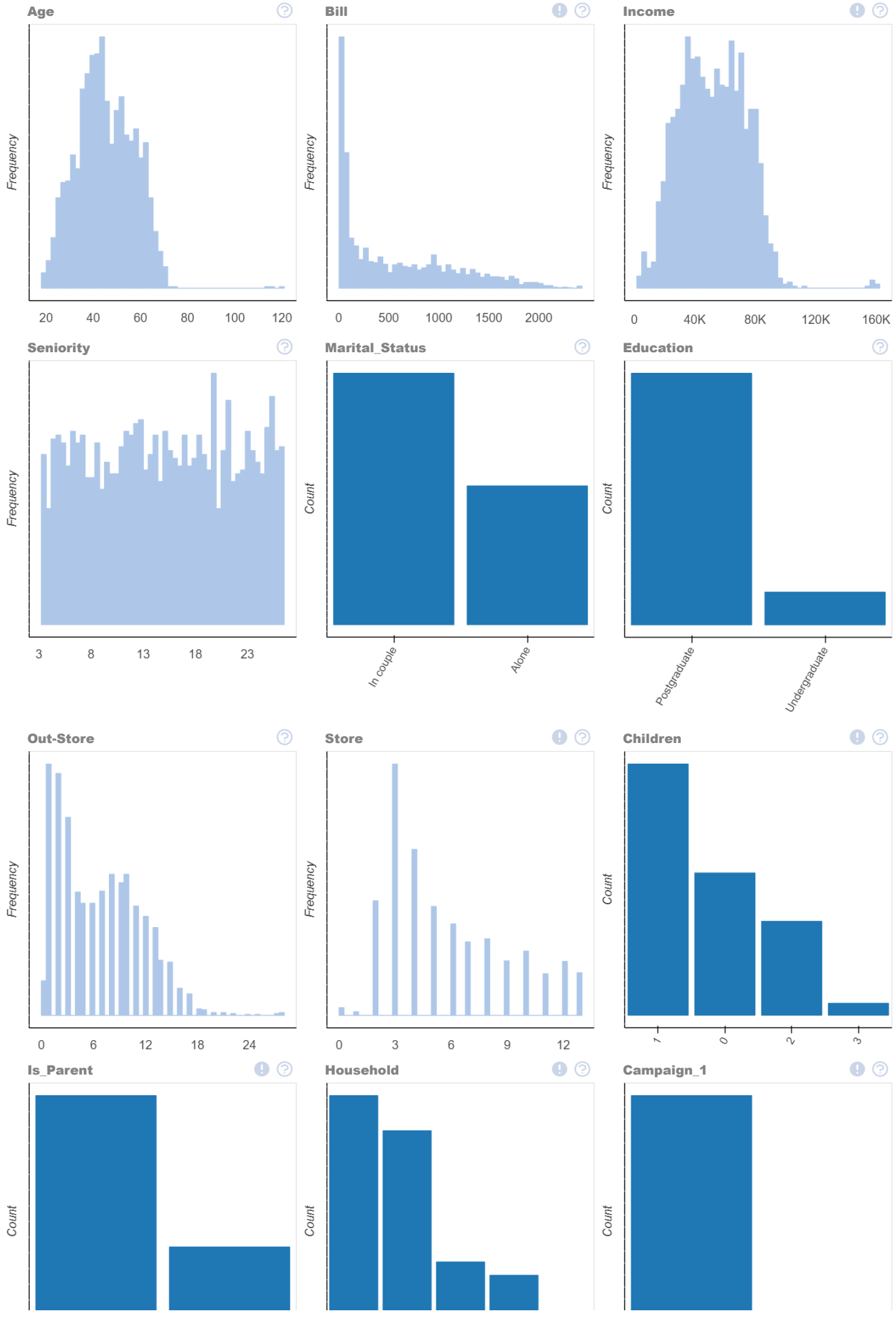
In [10]:
```
from dataprep.eda import plot, plot_correlation, plot_missing
```

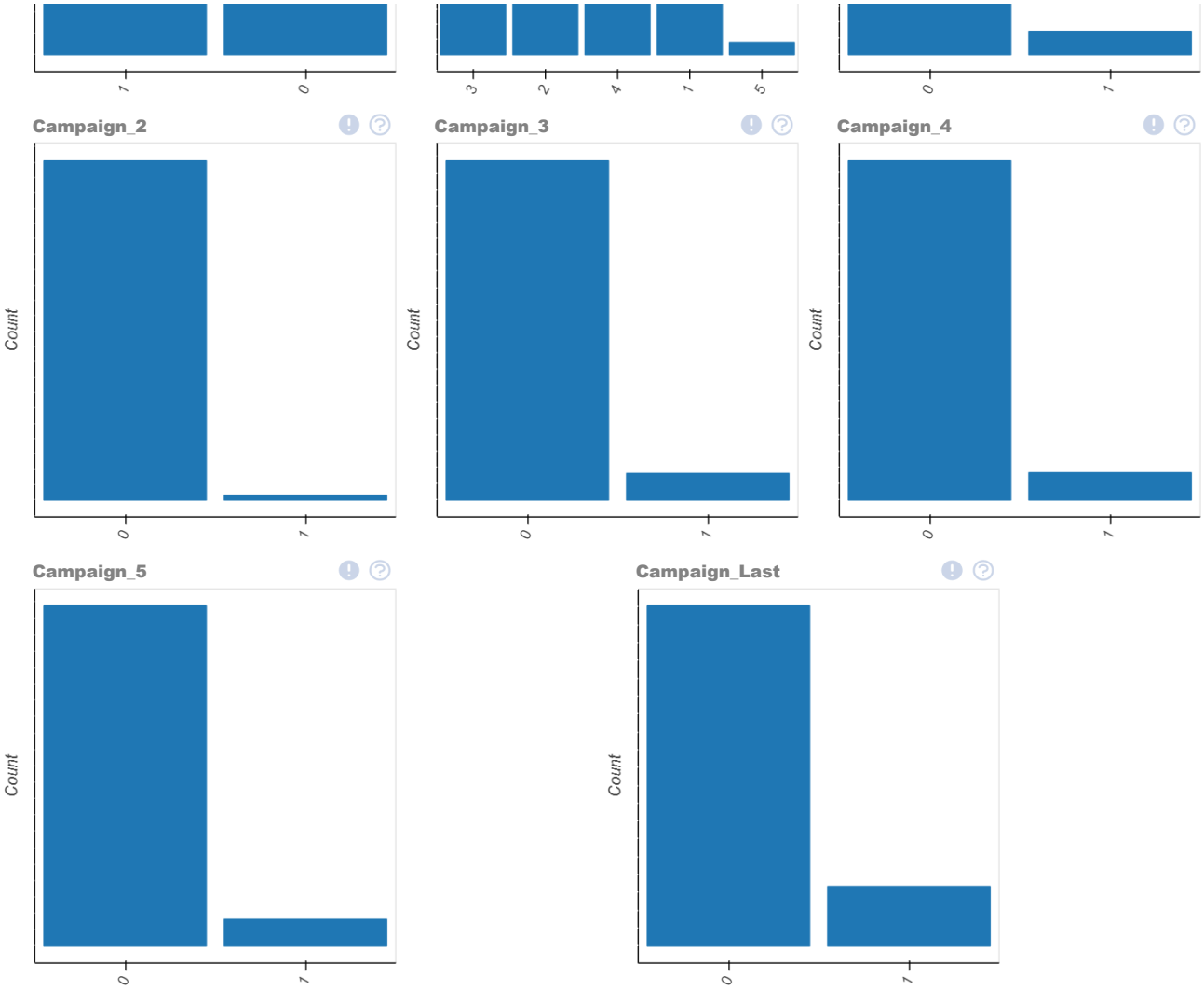
In [11]:
```
plot(df)
```

```
  0%|          | 0/652 [00:00<?, ?it/s]
```

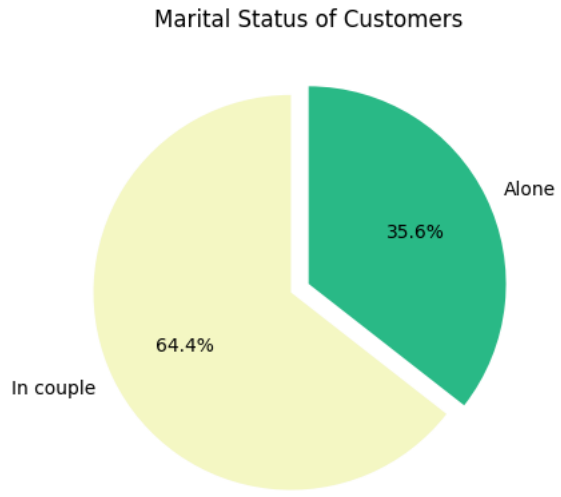
Out[11]:

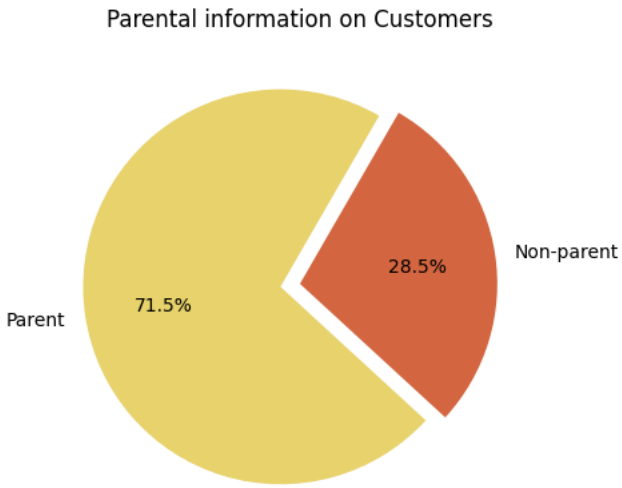
Show Stats and Insights

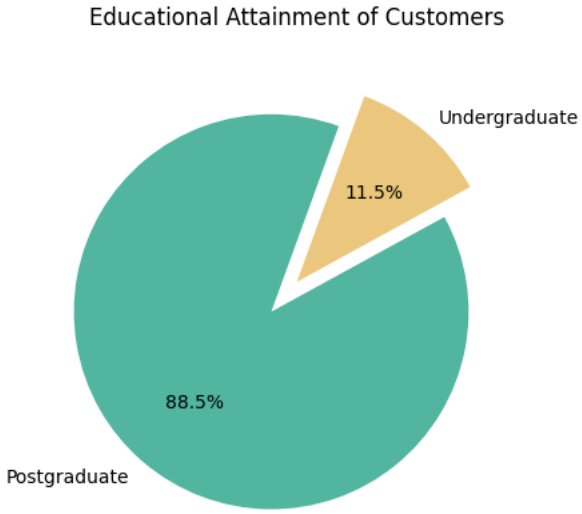




In [12]:
```
df['Marital_Status'].value_counts().plot(title= "Marital Status of Customers", ylabel = "", kind='pie', autopct = '%1.1f%%', explode=[0.1,0],
plt.show()
df['Is_Parent'].value_counts().plot(title= "Parental information on Customers", ylabel = "", kind='pie', autopct = '%1.1f%%', labels = {"Paren
plt.show()
df['Education'].value_counts().plot(title= "Educational Attainment of Customers", ylabel = "", kind='pie', autopct = '%1.1f%%', explode=[0.2,0
plt.show()
```







In [13]:
```
plot(df, 'Age')
```

```
  0%|          | 0/122 [00:00<?, ?it/s]
```

Out[13]:

Stats | Histogram | KDE Plot | Normal Q-Q Plot | Box Plot | Value Table

Overview

Approximate Distinct Count	59
Approximate Unique (%)	2.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	35824
Mean	45.1979
Minimum	18
Maximum	121
Zeros	0
Zeros (%)	0.0%
Negatives	0
Negatives (%)	0.0%

Quantile Statistics

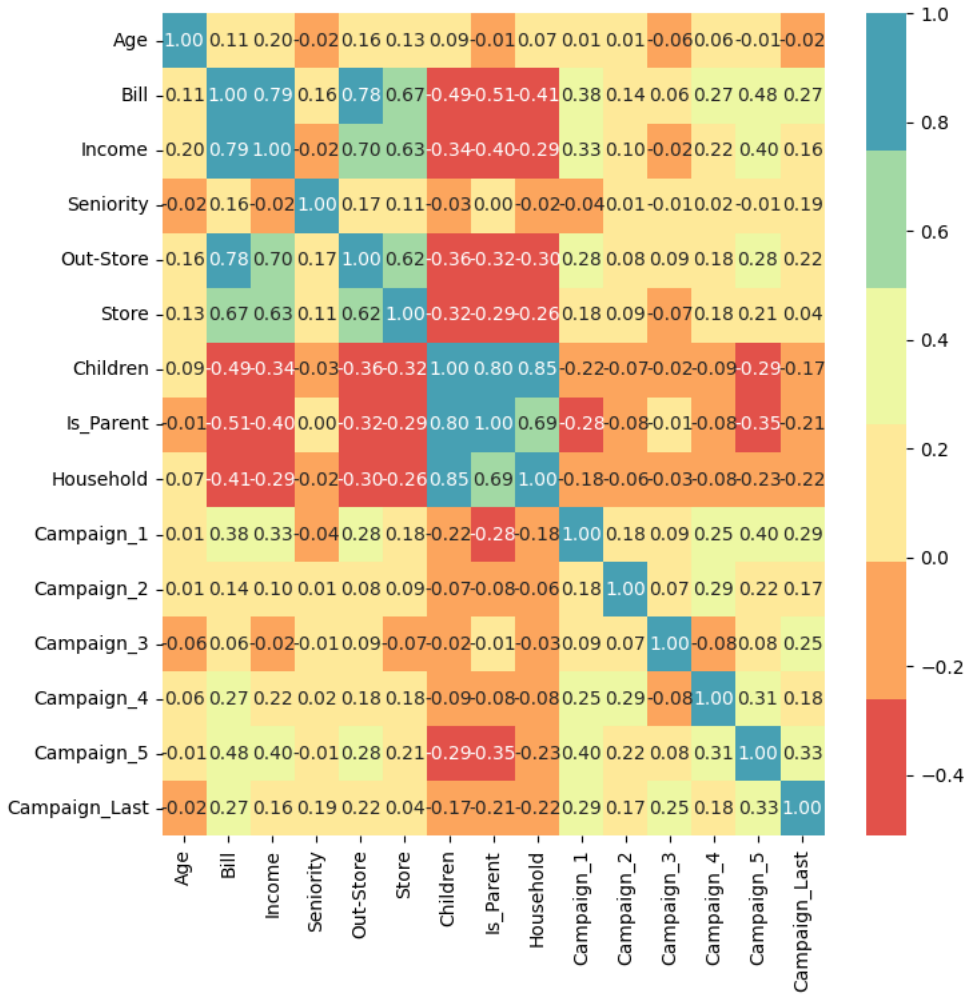
Minimum	18
5-th Percentile	26
Q1	37
Median	44
Q3	55
95-th Percentile	64
Maximum	121
Range	103
IQR	18

Descriptive Statistics

Mean	45.1979
Standard Deviation	11.9855
Variance	143.6521
Sum	101198
Skewness	0.349
Kurtosis	0.7126
Coefficient of Variation	0.2652

In [14]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(), cmap=sns.color_palette("Spectral"), annot=True, fmt='.2f')
```

Out[14]:
```
<AxesSubplot: >
```

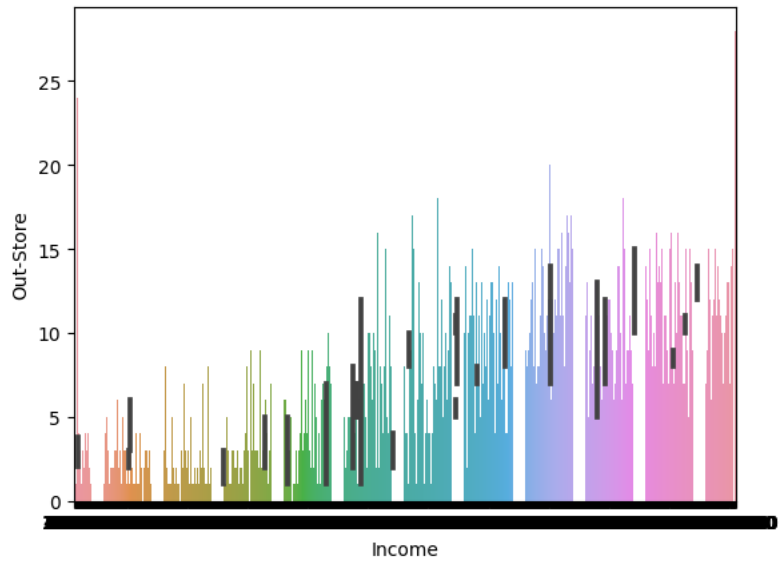


Observations

1. There is a strong positive correlation between out-store purchases and bill, and a moderately strong positive correlation between out-store purchases and incom
2. The correlations with income and with bill are stronger for out-store purchases than store purchases.
3. Majority of customers' age are between late 30's and mid-40's.

In [15]:
```
df.sort_values('Income', inplace=True)
sns.barplot(data =df, x='Income', y='Out-Store')
```

Out[15]:
```
<AxesSubplot: xlabel='Income', ylabel='Out-Store'>
```



Imputation using mean

I will be filling empty values in in the Income field using the mean. This is because normalization of the data will require no null values.

In [16]:
```
df['Income'].fillna(df['Income'].mean(), inplace=True)
```

Creating clusters

We will be making clusters of customers using their platforms used, income, and bill. Purchasing power will be operationalized by a composite of income and bill.

- **Cluster OH**: Customers with high out-store purchases, AND high income and high bill
- **Cluster IH**: Customers with low out-store purchases, AND high income and high bill
- **Cluster OL**: Customers with high out-store purchases, AND low income and low bill
- **Cluster IL**: Customers with low out-store purchases, AND low income and low bill.

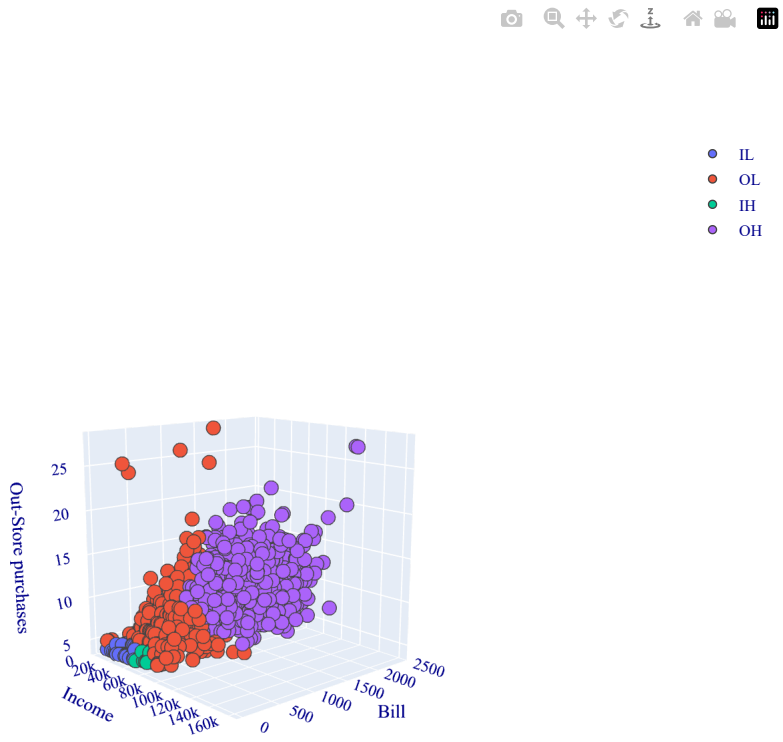
In [17]:
```
# Normalize the data
scaler = StandardScaler()
dfA_temp = df[['Out-Store','Income','Bill']]
X_std = scaler.fit_transform(dfA_temp)
X = normalize(X_std, norm= 'l2')

# Use Gaussian Mixture model to perform clustering
gmm = GaussianMixture(n_components=4, covariance_type='spherical',max_iter=2200, random_state=5).fit(X)
labels = gmm.predict(X)
dfA_temp['Cluster'] = labels
dfA_temp=dfA_temp.replace({0:'OH',1:'IL',2:'IH',3:'OL'})
data = df.merge(dfA_temp.Cluster, left_index=True, right_index=True)

summary=data[['Out-Store','Bill','Income','Cluster']]
summary.set_index("Cluster", inplace = True)
summary=summary.groupby('Cluster').describe().transpose()
summary.head()
```

Out[17]:

	Cluster	IH	IL	OH	OL
Out-Store	**count**	282.000000	693.000000	832.000000	432.000000
	mean	2.390071	2.437229	11.239183	7.858796
	std	1.423109	1.383490	3.077695	3.358124
	min	0.000000	0.000000	5.000000	0.000000
	25%	1.000000	1.000000	9.000000	6.000000

In [18]:
```
PLOT = go.Figure()
for C in list(dfA_temp.Cluster.unique()):
```

```
    PLOT.add_trace(go.Scatter3d(x = dfA_temp[dfA_temp.Cluster == C]['Income'],
                                y = dfA_temp[dfA_temp.Cluster == C]['Bill'],
                                z = dfA_temp[dfA_temp.Cluster == C]['Out-Store'],
                                mode = 'markers',marker_size = 6, marker_line_width = 1,
                                name = str(C)))
PLOT.update_traces(hovertemplate='Income %{x} <br>Bill: %{y} <br>Out-Store: %{z}')


PLOT.update_layout(width = 800, height = 800, autosize = True, showlegend = True,
                   scene = dict(xaxis=dict(title = 'Income', titlefont_color = '#00008B'),
                                yaxis=dict(title = 'Bill', titlefont_color = '#00008B'),
                                zaxis=dict(title = 'Out-Store purchases', titlefont_color = '#00008B')),
                   font = dict(family = "Serif", color  = '#00008B', size = 12))
```



Other observations

1. The highest income group has the largest variation in bill size.
2. The low purchasing power groups tend to stay below or within the 200 bill size.
3. Campaign with the best response was the last campaign.

In [ ]: