```
from google.colab import drive
drive.mount('/gdrive/')
%cd /gdrive
```

```
Mounted at /gdrive/
/gdrive
```

```
ls
```

```
MyDrive/  Shareddrives/
```

```
cd/gdrive/My Drive/Customer Personality/
```

```
/gdrive/My Drive/Customer Personality
```

```
ls
```

```
marketing_campaign.csv
```

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import preprocessing
sb.set()
```

```
data=pd.read_csv("marketing_campaign.csv",sep = '\t')
data
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custo
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2

```
data.info()
```

```
<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
```

Data preprocessing

```
def unique_columns(data):
    for i in _data.columns:
        print(i, ':', len(_data[i].unique()))
```

```
#Removing null values
data = data.dropna()
data.describe()
```

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	22
mean	5588.353339	1968.820397	52247.251354	0.441787	0.505415	49.012635	3
std	3249.376275	11.985554	25173.076661	0.536896	0.544181	28.948352	3
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	
25%	2814.750000	1959.000000	35303.000000	0.000000	0.000000	24.000000	
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	1
75%	8421.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	5
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	14

Removing Z Columns - because they contain constant variable and won't contribute to the analysis

```
data = data.drop(['Z_CostContact'], axis = 1)
data = data.drop(['Z_Revenue'], axis = 1)
```

```
data.head(10)
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014
5	7446	1967	Master	Together	62513.0	0	1	09-09-2013
6	965	1971	Graduation	Divorced	55635.0	0	1	13-11-2012
7	6177	1985	PhD	Married	33454.0	1	0	08-05-2013
8	4855	1974	PhD	Together	30351.0	1	0	06-06-2013
9	5899	1950	PhD	Together	5648.0	1	1	13-03-2014

Group Marrital Status

```
data['Marital_Status'].unique()
```

```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow', 'Alone',
       'Absurd', 'YOLO'], dtype=object)
```

```
data['Marital_Status'].value_counts()
```

```
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
YOLO          2
Absurd        2
Name: Marital_Status, dtype: int64
```

```
data_ms = data.copy()
```

```
data_ms['Marital_Status'] = data_ms['Marital_Status'].map({'Single':0, 'Together':
                                                           'Divorced':0, 'Widow':0
data_ms['Marital_Status'].unique()
```

```
array([0, 1])
```

```
data_ms['Marital_Status'].value_counts()
```

```
1    1430
0     786
Name: Marital_Status, dtype: int64
```

Grouping Education

```
data_ms['Education'].unique()
```

```
array(['Graduation', 'PhD', 'Master', 'Basic', '2n Cycle'], dtype=object)
```

```
data_ms['Education'].value_counts()
```

```
Graduation    1116
PhD            481
Master         365
2n Cycle       200
```

```
Basic          54
Name: Education, dtype: int64
```

```
data_ed = data_ms.copy()
```

Mapping Education into groups

```
data_ed['Education'] = data_ed['Education'].map({'Graduation':1, 'PhD':2, 'Master
data_ed['Education'].value_counts()
```

```
1    1116
2     846
0     254
Name: Education, dtype: int64
```

```
data_ed= data_ed.reset_index(drop=True)
data_dt = data_ed.copy()
```

Calculating Customers age

```
data_dt['Dt_Customer'] = pd.to_datetime(data_dt['Dt_Customer'], format='%d-%m-%Y')
```

```
data_dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2216 entries, 0 to 2215
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2216 non-null   int64
 1   Year_Birth           2216 non-null   int64
 2   Education            2216 non-null   int64
 3   Marital_Status       2216 non-null   int64
 4   Income               2216 non-null   float64
 5   Kidhome              2216 non-null   int64
 6   Teenhome             2216 non-null   int64
 7   Dt_Customer          2216 non-null   datetime64[ns]
 8   Recency              2216 non-null   int64
 9   MntWines             2216 non-null   int64
 10  MntFruits            2216 non-null   int64
 11  MntMeatProducts      2216 non-null   int64
 12  MntFishProducts      2216 non-null   int64
 13  MntSweetProducts     2216 non-null   int64
 14  MntGoldProds         2216 non-null   int64
 15  NumDealsPurchases    2216 non-null   int64
 16  NumWebPurchases      2216 non-null   int64
 17  NumCatalogPurchases  2216 non-null   int64
 18  NumStorePurchases    2216 non-null   int64
 19  NumWebVisitsMonth    2216 non-null   int64
 20  AcceptedCmp3         2216 non-null   int64
```

```
 21  AcceptedCmp4         2216 non-null   int64
 22  AcceptedCmp5         2216 non-null   int64
 23  AcceptedCmp1         2216 non-null   int64
 24  AcceptedCmp2         2216 non-null   int64
 25  Complain             2216 non-null   int64
 26  Response             2216 non-null   int64
dtypes: datetime64[ns](1), float64(1), int64(25)
memory usage: 467.6 KB
```

```
data_dt['Dt_Customer'][0].year
```

```
2012
```

```
data_dt['Year_Birth'][0]
```

```
1957
```

```
data_dt['Dt_Customer'].max()
```

```
Timestamp('2014-06-29 00:00:00')
```

```
int(data_dt['Dt_Customer'][10].year)
```

```
2012
```

```
Age=[]
```

```
print('As the Maximum value of customers joined year in 2014, So we shall assume t
```

```
As the Maximum value of customers joined year in 2014, So we shall assume that data was
```

```
for i in range(data_dt.shape[0]):
    a = int(2014) - data_dt['Year_Birth'][i]
    Age.append(a)
```

```
data_dt['Age']=Age
```

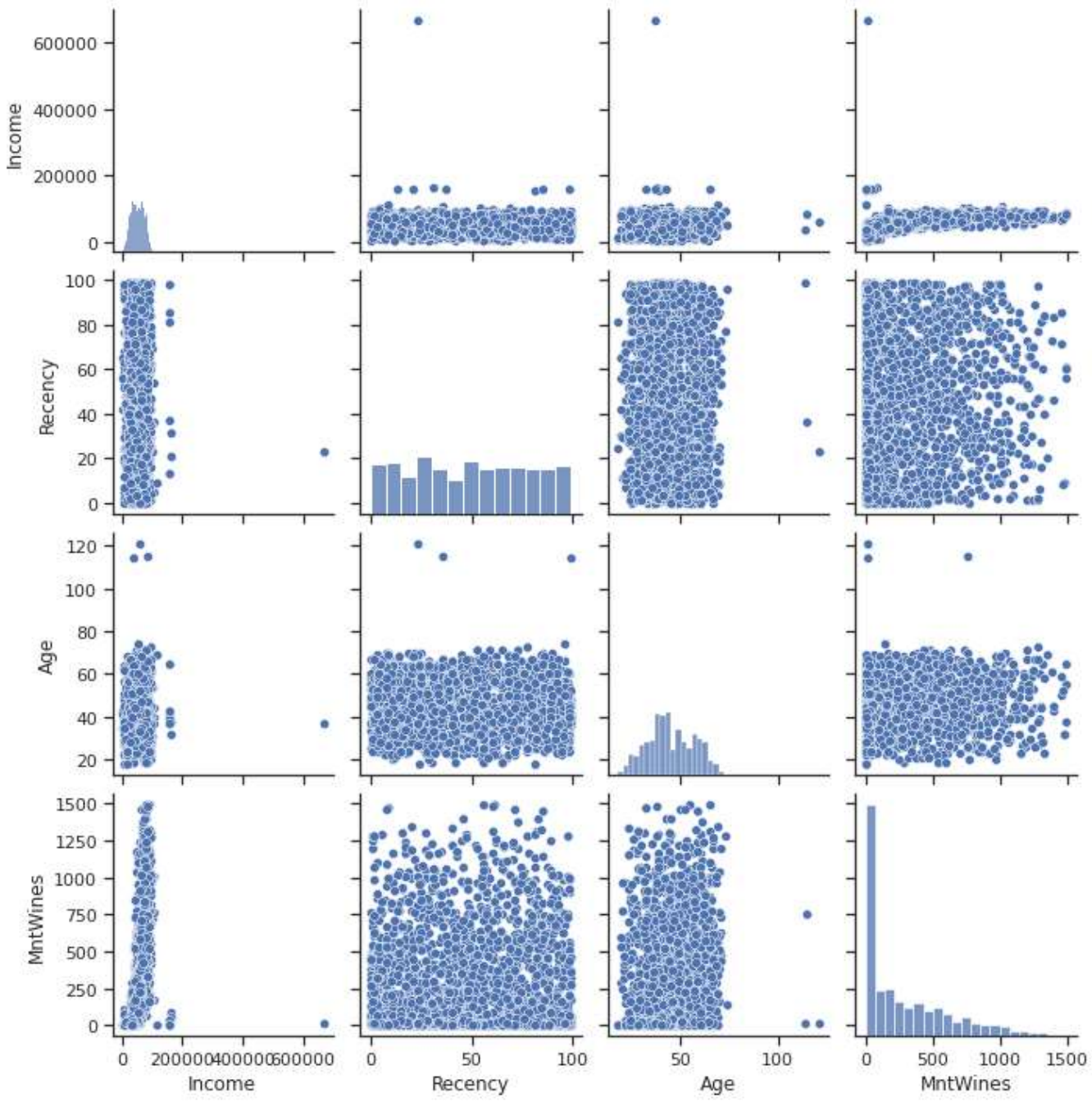
```
data_s = data_dt.copy()
```

Identifying Outliers and Data Distribution

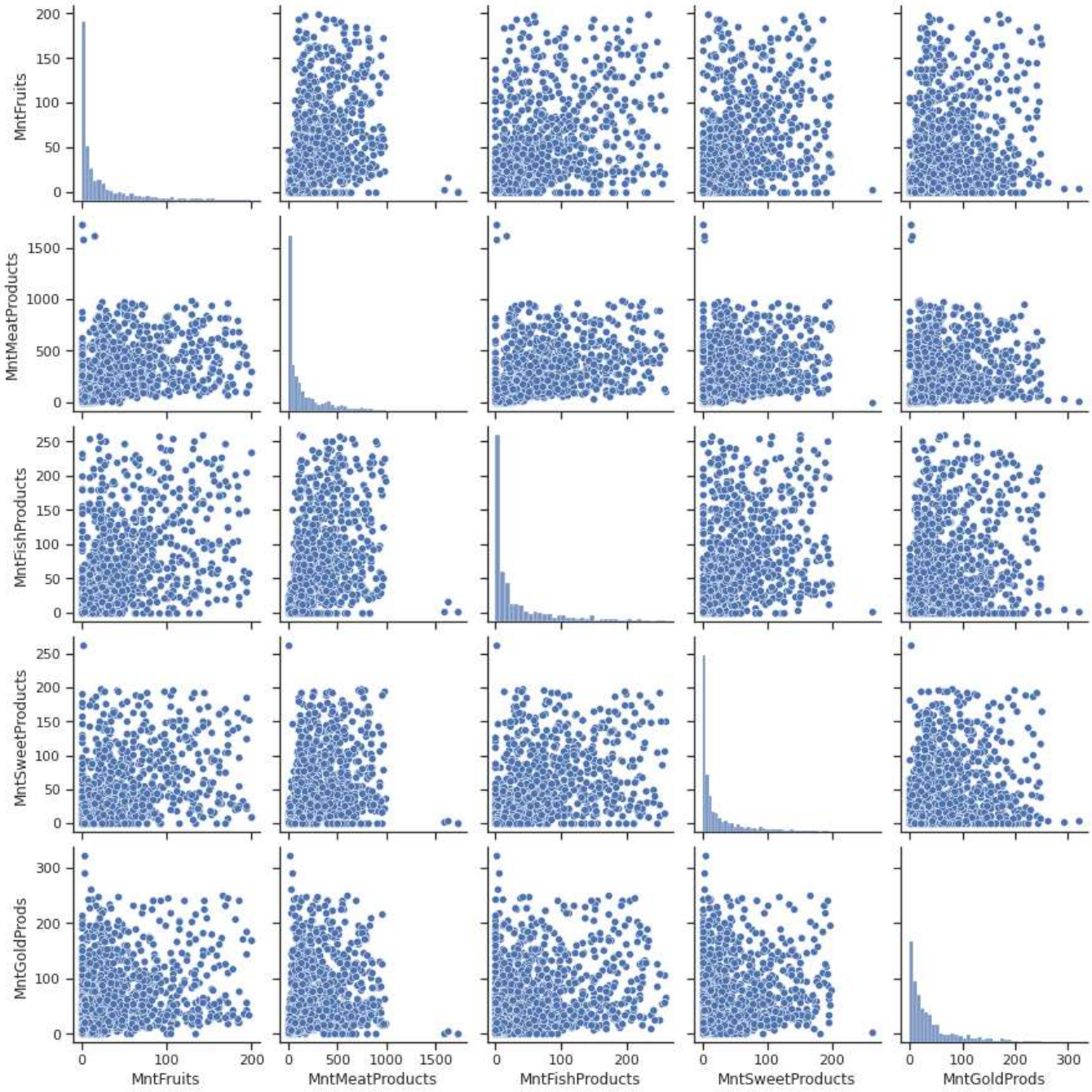
```
data_s['Income'].unique().max()
```

```
666666.0
```

```
sb.set(style="ticks", color_codes=True)
g = sb.pairplot(data_s, vars=["Income", "Recency", "Age", 'MntWines']) #, hue="Clu
plt.show()
```

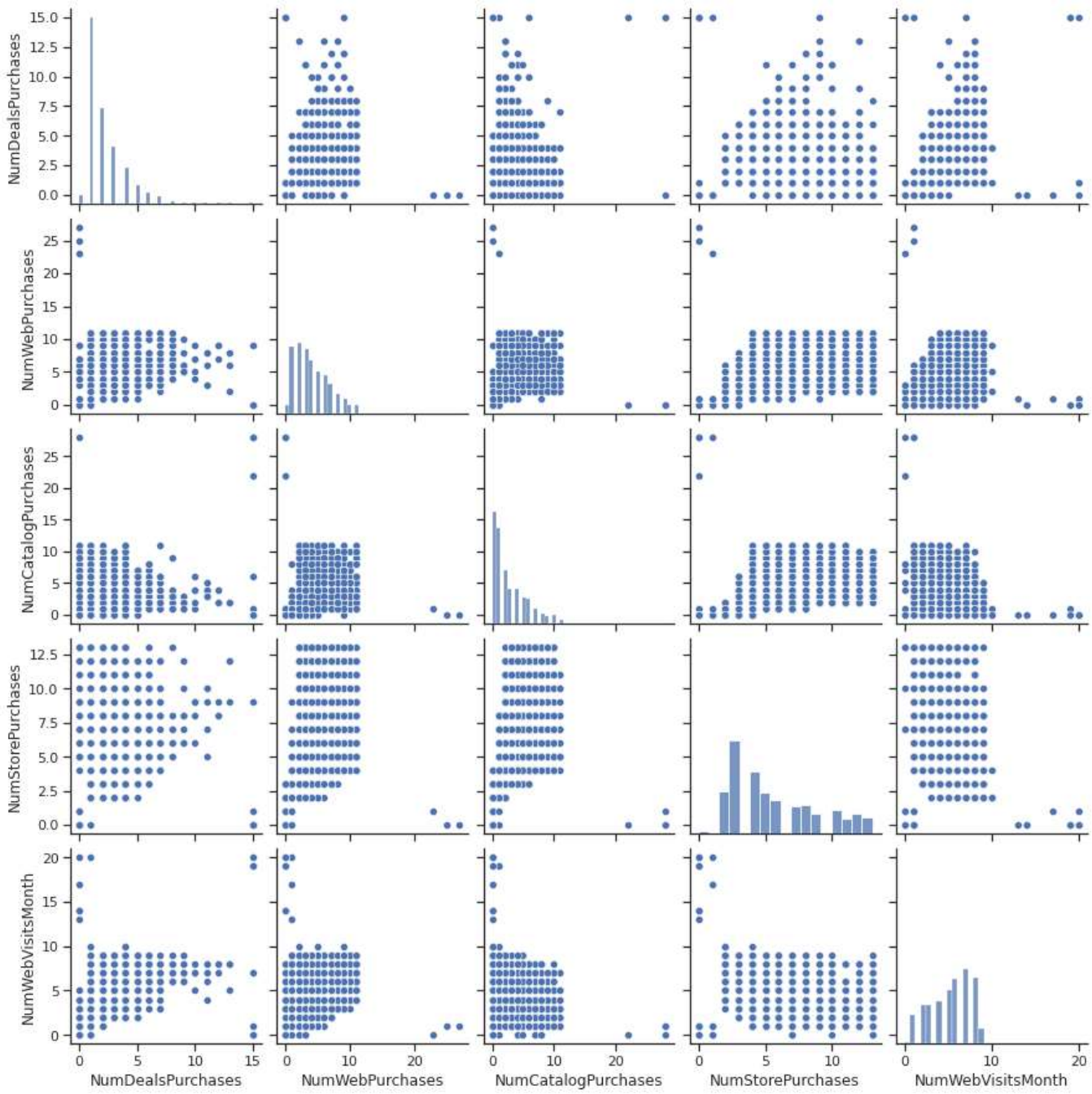



```
sb.set(style="ticks", color_codes=True)
g = sb.pairplot(data_s, vars=['MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds'])
plt.show()
```




```
sb.set(style="ticks", color_codes=True)
g = sb.pairplot(data_s, vars=['NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth'])
```

```
plt.show()
print('We will remove certain outliers that are prone to distorted the results')
```




```
We will remove certain outliers that are prone to distorted the results
```

Removing Outliers

We will remove following values

1. Ages greater than 100
2. Income greater than 60,000
3. MntMeatProducts greater than 1500
4. MntSweetProducts greater than 250
5. NumWebPurchases greater than 20
6. NumCatalogPurchases greater than 20

```
data_s = data_s[data_s['Age'] < 100]
data_s = data_s[data_s['Income'] < 60000]
data_s = data_s[data_s['MntMeatProducts'] < 1500]
data_s = data_s[data_s['MntSweetProducts'] < 250]
data_s = data_s[data_s['MntGoldProds'] < 300]
data_s = data_s[data_s['NumWebPurchases'] < 20]
data_s = data_s[data_s['NumCatalogPurchases'] < 20]
```

```
data_s = data_s.reset_index(drop=True)
```

```
data_ags = data_s.copy()
```

```
column_names_reordered = ['ID', 'Year_Birth','Age', 'Education', 'Marital_Status',
       'Kidhome', 'Teenhome', 'Dt_Customer', 'Recency', 'MntWines',
       'MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth', 'AcceptedCmp3', 'AcceptedCmp4',
       'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Complain',
       'Response']
```

```
data_ag = data_ags[column_names_reordered]
```

```
data_ags.columns.values
```

```
array(['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',
       'Response', 'Age'], dtype=object)
```

```
data_ag.head(5)
```

	ID	Year_Birth	Age	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cust
0	5524	1957	57	1	0	58138.0	0	0	2012-(
1	2174	1954	60	1	0	46344.0	1	1	2014-(
2	6182	1984	30	1	1	26646.0	1	0	2014-(
3	5324	1981	33	2	1	58293.0	1	0	2014-(
4	965	1971	43	1	0	55635.0	0	1	2012-1

Grouping Kidhome, Teenhome, Spending and Purchases

```
data_ag['Kids'] = data_ag['Kidhome'] + data_ag['Teenhome']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  """Entry point for launching an IPython kernel.
```

```
data_ag['Total Spending'] = (data_ag['MntWines'] + data_ag['MntFruits'] +
                             data_ag['MntMeatProducts'] + data_ag['MntFishProducts
                             data_ag['MntSweetProducts'] + data_ag['MntGoldProds']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
data_ag['Total Purchases'] = (data_ag['NumDealsPurchases'] + data_ag['NumWebPurcha
                              data_ag['NumCatalogPurchases'] + data_ag['NumStorePur
                              data_ag['NumWebVisitsMonth'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
data_ag = data_ag.drop(['Kidhome', 'Teenhome', 'ID', 'Year_Birth', 'Dt_Customer'],
```

```
data_tp = data_ag.drop(['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProduc
       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth'], axis=1)
```

Analysis from a Wider Prespective (Considering Total Spending and Total Purchases)

```
data_tp.columns.values
```

```
array(['Age', 'Education', 'Marital_Status', 'Income', 'Recency',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Response', 'Kids', 'Total Spending',
       'Total Purchases'], dtype=object)
```

```
columns_tp = ['Income', 'Recency', 'Total Spending', 'Total Purchases']
```

```
columns_tp
```

```
['Income', 'Recency', 'Total Spending', 'Total Purchases']
```

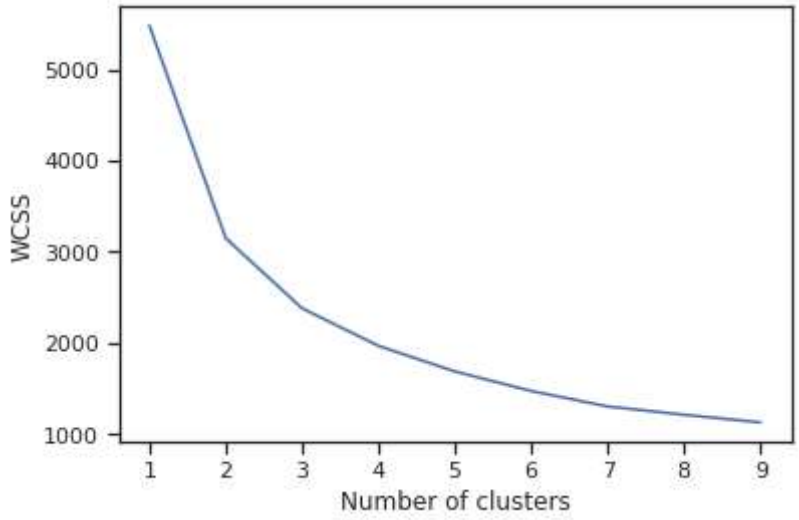
```
data_tbs = data_tp[columns_tp]
# scale the data for better results
x_scaled_tbs = preprocessing.scale(data_tbs)
x_scaled_tbs
```

```
array([[ 1.52718002,  0.31214897,  4.85806776,  2.10397331],
       [ 0.6155132 , -0.38445323, -0.73596908, -0.97618709],
       [-0.90712653, -0.80241455, -0.64449427, -0.53616417],
       ...,
       [-0.30612522,  1.1132415 , -0.7254143 , -0.97618709],
       [ 1.43774484,  1.4615426 ,  3.53520119,  1.07725318],
       [ 1.11989051, -0.31479301, -0.22582107,  0.05053304]])
```

```
# Createa an empty list
wcss =[]
for i in range(1,10):
    # Clsuter solution with i clusters
    kmeans = KMeans(i)
    # Fit the STANDARDIZED data
    kmeans.fit(x_scaled_tbs)
```

```
    kmeans.fit(x_scaled_tbs)
    # Append the WCSS for the iteration
    wcss.append(kmeans.inertia_)
# Check the result
# Plot the number of clusters vs WCSS
plt.plot(range(1,10),wcss)
# Name your axes
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
print('From the graph we shall consider 4 clusters optimal number, as we try to mi
```

```
From the graph we shall consider 4 clusters optimal number, as we try to minimize the WC
```



Modeling - KMeans

```
kmeans_scaled_tbs = KMeans(4)
kmeans_scaled_tbs.fit(x_scaled_tbs)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
identified_clusters_tbss = kmeans_scaled_tbs.fit_predict(x_scaled_tbs)
identified_clusters_tbss
```

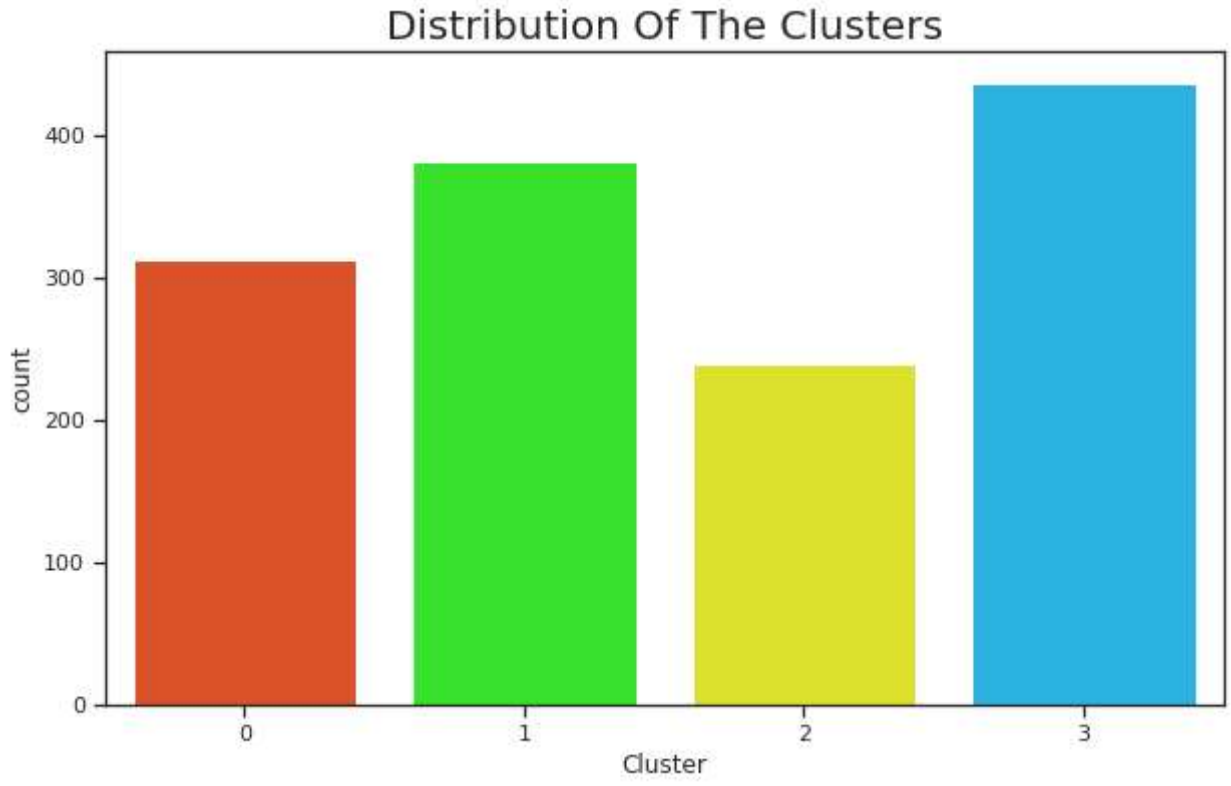
```
array([2, 0, 1, ..., 3, 2, 0], dtype=int32)
```

```
data_tbss_clusters = data_tp.copy()
data_tbss_clusters['Cluster'] = identified_clusters_tbss
data_tbss_clusters.head(6)
```

	Age	Education	Marital_Status	Income	Recency	AcceptedCmp3	AcceptedCmp4	Accepte
0	57	1	0	58138.0	58	0	0	
1	60	1	0	46344.0	38	0	0	
2	30	1	1	26646.0	26	0	0	
3	33	2	1	58293.0	94	0	0	

Model Evaluation

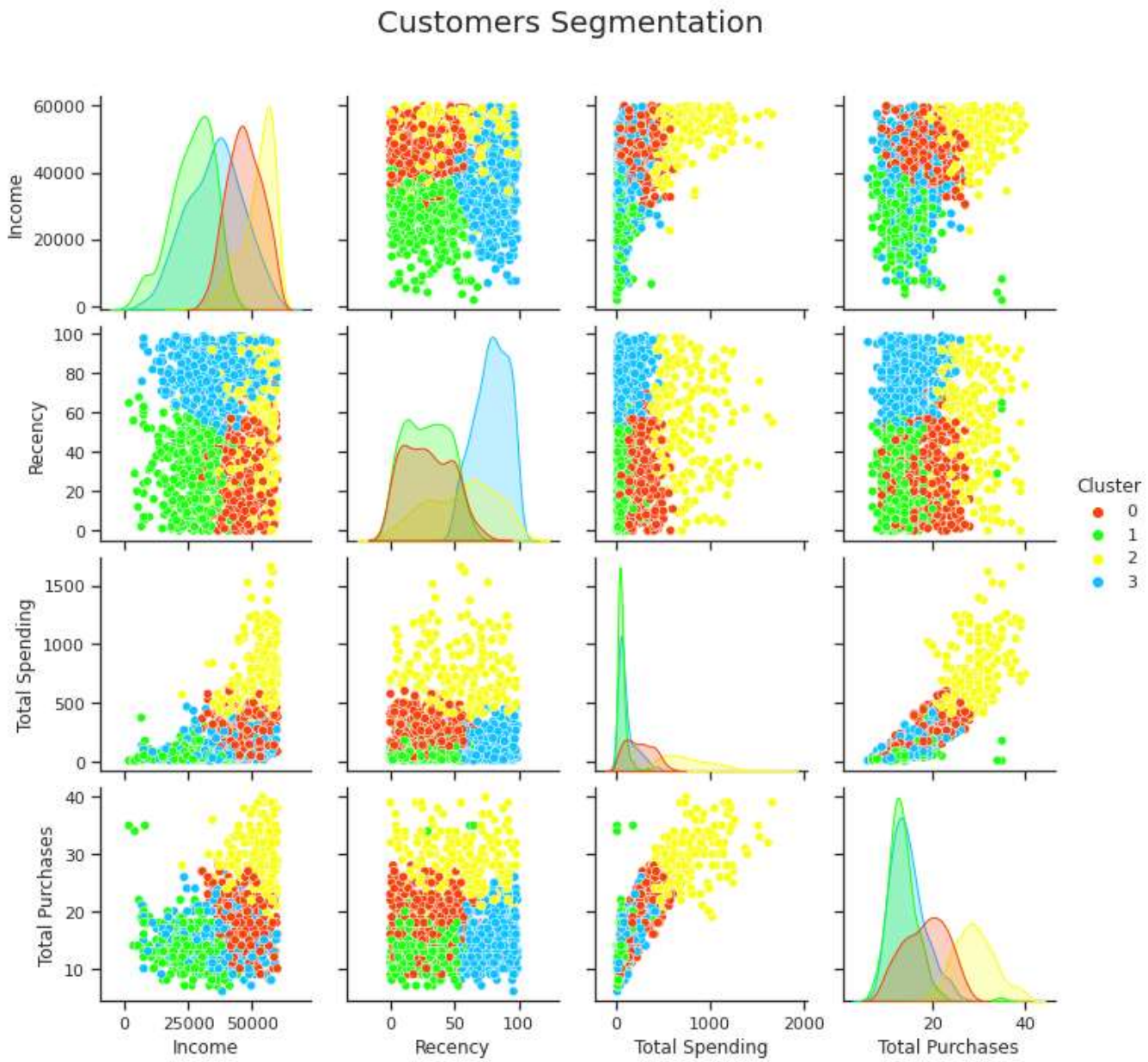
```
plt.figure(figsize=(10,6))
Cl = sb.countplot(x = 'Cluster', data = data_tbss_clusters, palette= ['#F7420D',
Cl.set_title("Distribution Of The Clusters", size = 20)
plt.show()
```



Customer Segmentation

```
Features = ['Income', 'Recency', 'Total Spending', 'Total Purchases', 'Cluster']
plt.figure()
sb.pairplot(data_tbss_clusters[Features], hue = "Cluster", palette=['#F7420D', '#
plt.subplots_adjust(top=0.9)
plt.suptitle('Customers Segmentation', size = 20)
plt.show()
```

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




```
data_cs = {'Features': ['Income', 'Spending', 'Purchases', 'Recency', 'Customer Se
           'Cluster 0': ['High', 'High', 'High', 'High', 'Impulsive'], 'Cluster 1
          , 'Cluster 2': ['Avg', 'Low', 'Low', 'Low', 'Wandering'], 'Cluster 3': |
cs = pd.DataFrame(data_cs)
cs
```

	Features	Cluster 0	Cluster 1	Cluster 2	Cluster 3
0	Income	High	Low	Avg	High
1	Spending	High	Low	Low	Avg

Clusters Distribution Along the Features

```
data_tbss_clusters.columns.values
```

```
array(['Age', 'Education', 'Marital_Status', 'Income', 'Recency',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Response', 'Kids', 'Total Spending',
       'Total Purchases', 'Cluster'], dtype=object)
```

```
grouped = (data_tbss_clusters.groupby(['Cluster']).agg({'Marital_Status': 'sum',
                                                        'Education': 'sum', 'Recer
                                                        'Income': 'sum','Total Spe
grouped
```

Cluster	Marital_Status	Age	Kids	Education	Recency	Income	Total Spending	Tota Purchase
0	200	14674	407	436	8985	14669715.0	76170	576
1	249	15228	417	421	10786	10065084.0	22101	513
2	152	11379	260	323	13157	12460150.0	179218	694
3	288	18959	554	513	34254	15387382.0	46085	635

```
percents_df = grouped.apply(lambda x: round((x/x.sum()*100),2))
percents_df
```

Cluster	Marital_Status	Age	Kids	Education	Recency	Income	Total Spending	Total Purchases
0	22.50	24.36	24.85	25.75	13.37	27.90	23.54	23.82
1	28.01	25.28	25.46	24.87	16.05	19.14	6.83	21.21
2	17.10	18.89	15.87	19.08	19.58	23.70	55.39	28.71
3	32.40	31.47	33.82	30.30	50.99	29.26	14.24	26.26

```
plt.figure(figsize=(15,8))
s = sb.barplot(x = 'Features', y = 'Percentage', hue = 'Cluster',
               data = percents_df.stack().reset_index().rename(columns = {'level_1':
```

```
s.set_title("Distribution Of Clusters along the Features", size=20)
```

```
Text(0.5, 1.0, 'Distribution Of Clusters along the Features')
```

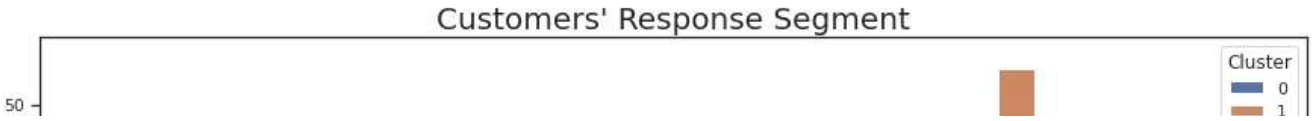
Clusters Distribution over Customers Interaction

```
plt.figure(figsize=(15,8))

grouped = (data_tbss_clusters.groupby(['Cluster']).agg({'AcceptedCmp1': 'sum','Acc
                                                        'AcceptedCmp3': 'sum','Acc
                                                        'AcceptedCmp5': 'sum', 'Re
           .stack().reset_index().rename(columns = {'level_1':'weights', 0: 'value

h = sb.barplot(x = 'weights', y = 'value', hue = 'Cluster', data = grouped)
h.set_title("Customers' Response Segment", size=20)
```

```
Text(0.5, 1.0, "Customers' Response Segment")
```



Result

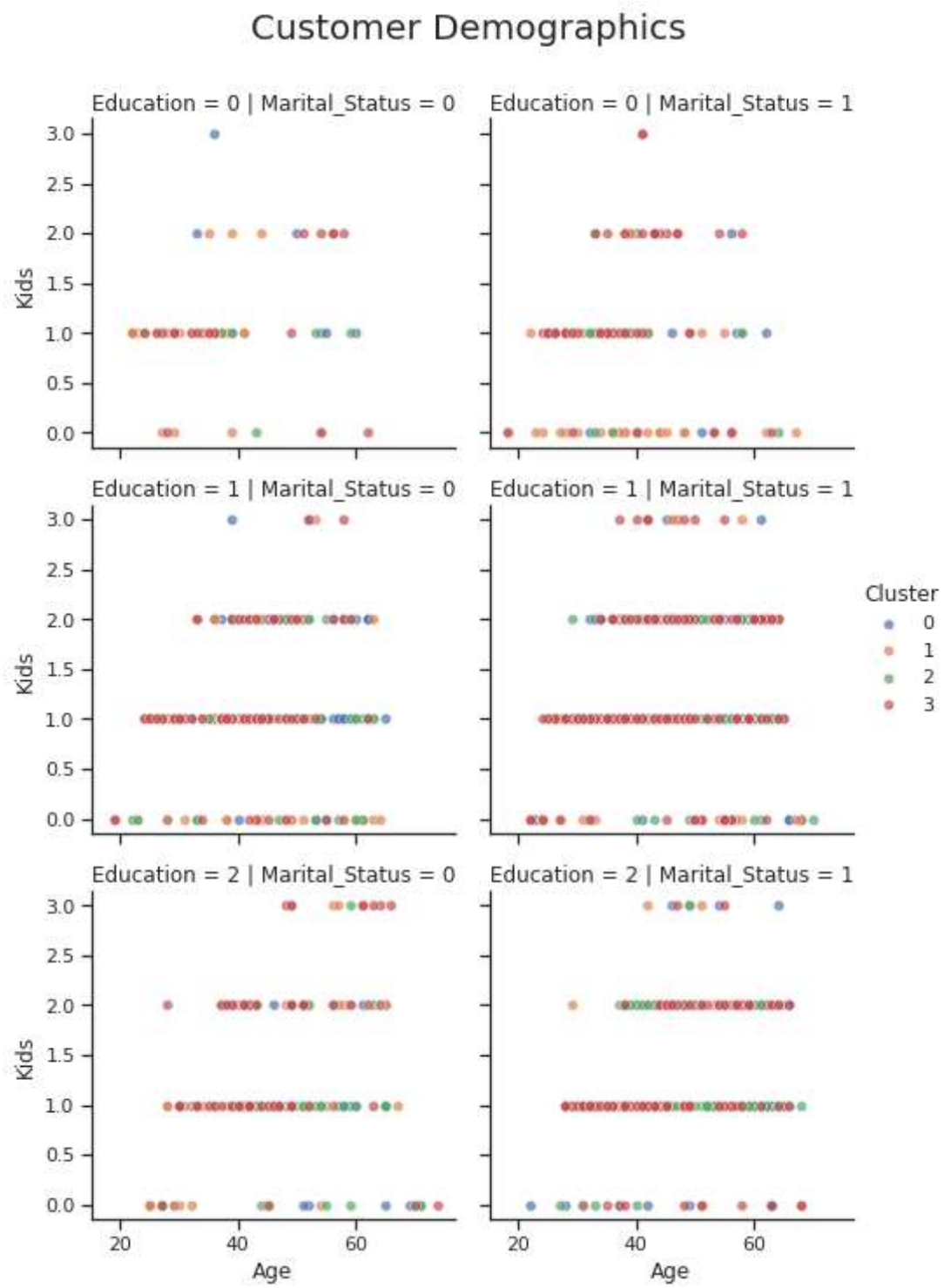

```
plt.figure(figsize=(15,8))
pl = sb.countplot(x = 'Age', hue=data_tbss_clusters['Cluster'], data = data_tbss_c
pl.set_title("Age Distribution", size=20)
plt.show()
```

```
plt.figure(figsize=(15,8))
pl = sb.countplot(x = 'Kids', hue=data_tbss_clusters['Cluster'], data = data_tbss_
pl.set_title("Distribution Of The Clusters", size=20)
plt.show()
```

```
sb.scatterplot(data = data_tbss_clusters, x = 'Age', y = 'Income', hue='Cluster',
```

```
g = sb.FacetGrid(data_tbss_clusters, col="Marital_Status", row="Education", hue="C
g.map(sb.scatterplot, "Age", "Kids", alpha=.7)
g.add_legend()
g.fig.subplots_adjust(top=0.9) # adjust the Figure in rp
g.fig.suptitle('Customer Demographics', size=20)
```

```
Text(0.5, 0.98, 'Customer Demographics')
```




```
data_cs = {'Features': ['Income', 'Spending', 'Purchases', 'Recency', 'Customer Se
           'Cluster 0': ['High', 'High', 'High', 'High', 'Impulsive'], 'Cluster 1
           'Cluster 2': ['Avg', 'Low', 'Low', 'Low', 'Wandering'], 'Cluster 3': [
```

```
, 'Cluster 2' : [ 'Avg' , 'Low' , 'Low' , 'Low' , 'Wandering' ], 'Cluster 3' : [
cs = pd.DataFrame(data_cs)
cs
```

	Features	Cluster 0	Cluster 1	Cluster 2	Cluster 3
0	Income	High	Low	Avg	High
1	Spending	High	Low	Low	Avg
2	Purchases	High	Low	Low	Avg
3	Recency	High	Low	Low	Low
4	Customer Segments	Impulsive	Need-Based	Wandering	Loyal