In [1]:

```python
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
```

In [2]:

```python
import csv
data = pd.read_csv('marketing_campaign.csv', sep='\t')
data.head()
```

Out[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	AcceptedCmp3
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	7	0
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	5	0
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...	4	0
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...	6	0
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...	5	0

5 rows × 29 columns

In [3]:

```python
data.describe()
```

Out[3]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	...
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	...
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.950000	37.525446	...
std	3246.662198	11.984069	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	...
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	16.000000	3.000000	...
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	...
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	...
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	...

8 rows × 26 columns

In [4]:

```python
data.isna().sum()
```

Out[4]:

```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                 24
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
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
dtype: int64
```

In [5]:

```python
#preprocessing (Drop NA values)
df = data[~data['Income'].isna()]
```

```
df.isna().sum()
```

Out[5]:

```
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
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

In [6]:

```
df.columns
```

Out[6]:

```
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')
```

In [7]:

```
import matplotlib.pyplot as plt
import seaborn as sns
print(df['Year_Birth'].unique())
```

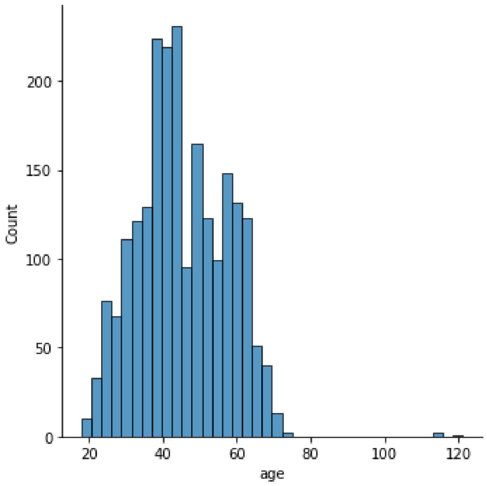
```
[1957 1954 1965 1984 1981 1967 1971 1985 1974 1950 1976 1959 1952 1987
 1946 1980 1949 1982 1979 1951 1969 1989 1963 1970 1973 1943 1975 1996
 1968 1964 1977 1978 1955 1966 1988 1948 1958 1972 1960 1983 1945 1991
 1962 1953 1956 1992 1961 1900 1986 1893 1990 1947 1899 1993 1994 1941
 1944 1995 1940]
```

In [8]:

```
#Lets say data is collected on 07-12-2014
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], utc=False)
last_enrollment = pd.to_datetime(df['Dt_Customer'].max(), utc=False)
df['Days_Enrolled'] = (last_enrollment-df['Dt_Customer']).dt.days
df.drop(columns = ['Dt_Customer'], inplace=True)
df['ActiveDays'] = df['Days_Enrolled'] - df['Recency']
```

In [9]:

```
df['age'] = 2014 - df['Year_Birth']
sns.displot(df['age'])
#drop dob column
df.drop(['Year_Birth'],axis=1,inplace=True)
```



In [10]:

```
print(df['Education'].unique())
#Education column
df['Education'] = df['Education'].replace(['PhD','Master','2n Cycle'],'Post Graduate')
df['Education'] =df['Education'].replace(['Graduation','Basic'],'Under Graduate')
df['Education'].unique()
```

```
['Graduation' 'PhD' 'Master' 'Basic' '2n Cycle']
```

Out[10]:

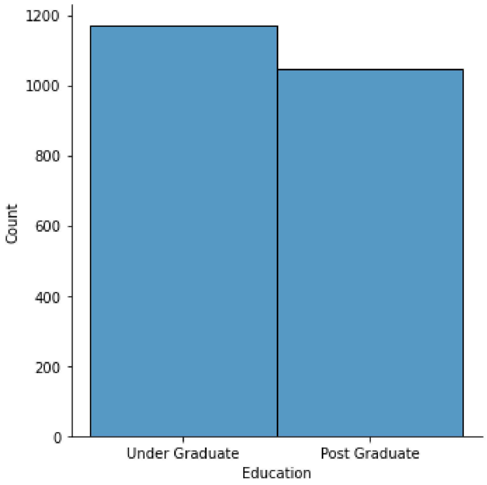
```
array(['Under Graduate', 'Post Graduate'], dtype=object)
```

In [11]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.displot(df['Education'])
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x228ff4a8700>
```



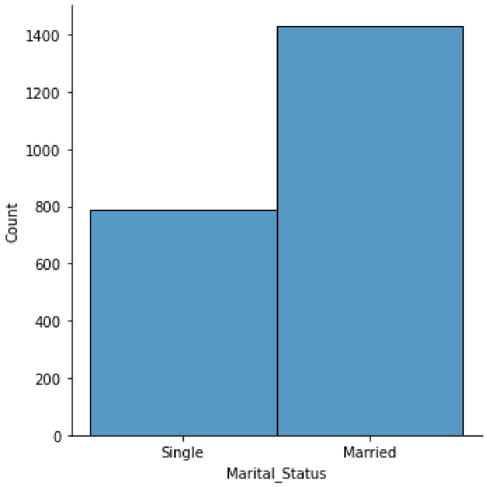
In [12]:

```
# maritial status
print(df['Marital_Status'].unique())
df['Marital_Status'] = df['Marital_Status'].replace(['Together','Married'],'Married')
df['Marital_Status'] = df['Marital_Status'].replace(['Single','Divorced','Widow','Alone','Absurd','YOLO'],'Single')
print(df['Marital_Status'].unique())
sns.displot(df['Marital_Status'])
```

```
['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Alone' 'Absurd' 'YOLO']
['Single' 'Married']
```

Out[12]:

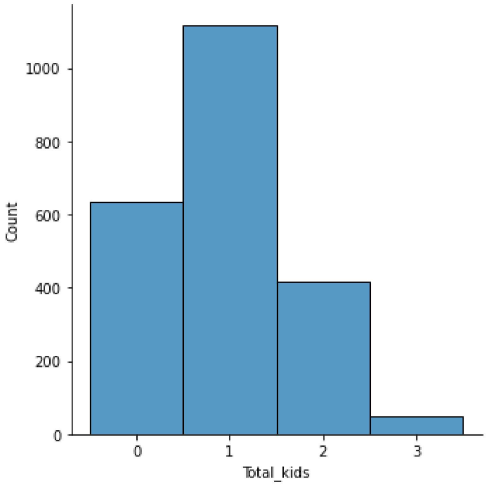
```
<seaborn.axisgrid.FacetGrid at 0x228ff209820>
```



In [13]:

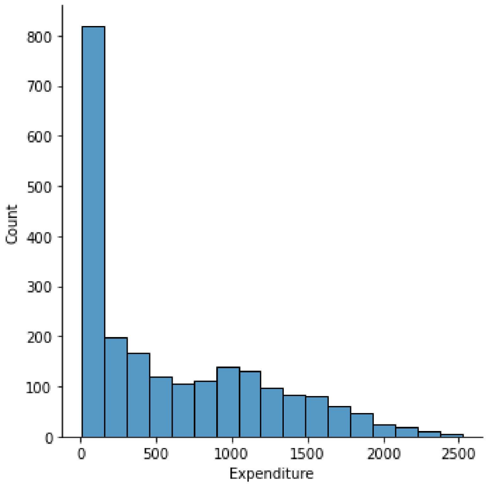
```
# Combining features of children
print(df['Kidhome'].unique())
print(df['Teenhome'].unique())
df['Total_kids'] = df['Kidhome'] + df['Teenhome']
print(df['Total_kids'].unique())
sns.displot(df['Total_kids'].sort_values().astype(str))

#drop kidhome, Teenhome
df.drop(['Kidhome', 'Teenhome'],axis=1,inplace=True)
```

```
[0 1 2]
[0 1 2]
[0 2 1 3]
```



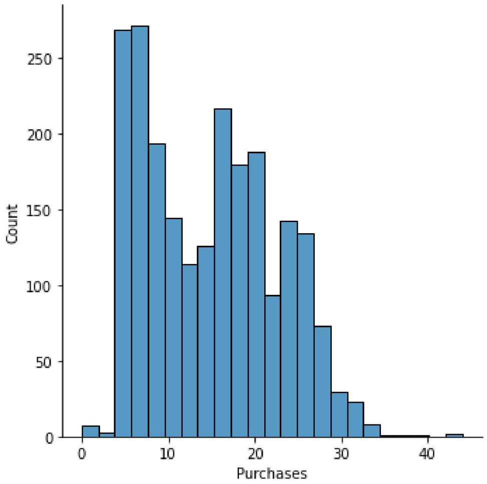
In [14]:

```
# Creating new features using expense
df['Expenditure'] = df['MntWines']+df['MntFruits']+df['MntMeatProducts']+df['MntFishProducts']+df['MntSweetProducts']+df['MntGoldPro
sns.displot(df['Expenditure'])

#drop columns
df.drop(['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds'],axis=1,inplace=True)
```



In [15]:

```
df['Purchases'] = df['NumDealsPurchases'] + df['NumWebPurchases'] + df['NumCatalogPurchases'] + df['NumStorePurchases']
sns.displot(df['Purchases'])


#drop columns
df.drop(['NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases'],axis=1,inplace=True)
```



In [16]:

```
df['TotalAcceptedCmp'] = df['AcceptedCmp1']+df['AcceptedCmp2']+df['AcceptedCmp3']+df['AcceptedCmp4']+df['AcceptedCmp5']+df['Response

#drop columns
df.drop(['AcceptedCmp1','AcceptedCmp2','AcceptedCmp3','AcceptedCmp4','AcceptedCmp5','Response'],axis=1,inplace=True)
```

In [17]:

```
df.columns
```

Out[17]:

```
Index(['ID', 'Education', 'Marital_Status', 'Income', 'Recency',
       'NumWebVisitsMonth', 'Complain', 'Z_CostContact', 'Z_Revenue',
       'Days_Enrolled', 'ActiveDays', 'age', 'Total_kids', 'Expenditure',
       'Purchases', 'TotalAcceptedCmp'],
      dtype='object')
```

In [18]:

```
df.drop(['ID','Z_CostContact', 'Z_Revenue'], axis = 1, inplace = True)
df.columns
```

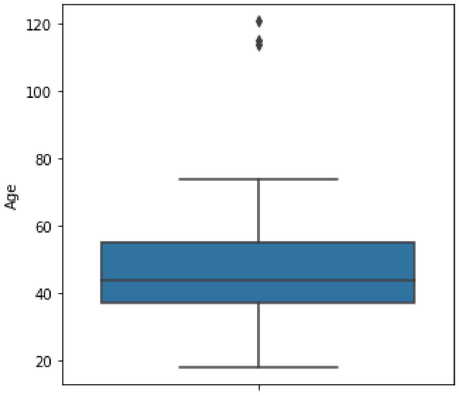
Out[18]:

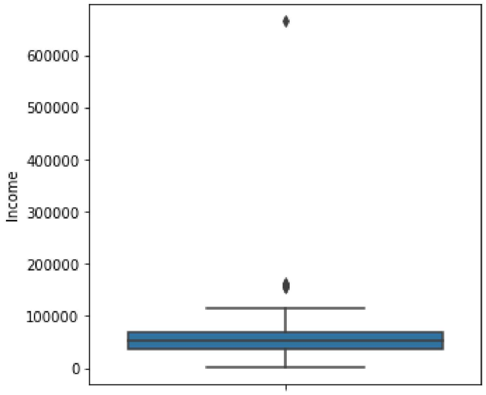
```
Index(['Education', 'Marital_Status', 'Income', 'Recency', 'NumWebVisitsMonth',
       'Complain', 'Days_Enrolled', 'ActiveDays', 'age', 'Total_kids',
       'Expenditure', 'Purchases', 'TotalAcceptedCmp'],
      dtype='object')
```

In [19]:

```
#Removing outliers

plt.figure(figsize=(5,5))
sns.boxplot(y=df.age);
plt.ylabel('Age');


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

```
sns.boxplot(y=df.Income);
plt.ylabel('Income');
```





In [20]:

```
df = df.query("Purchases != 0")
```

In [21]:

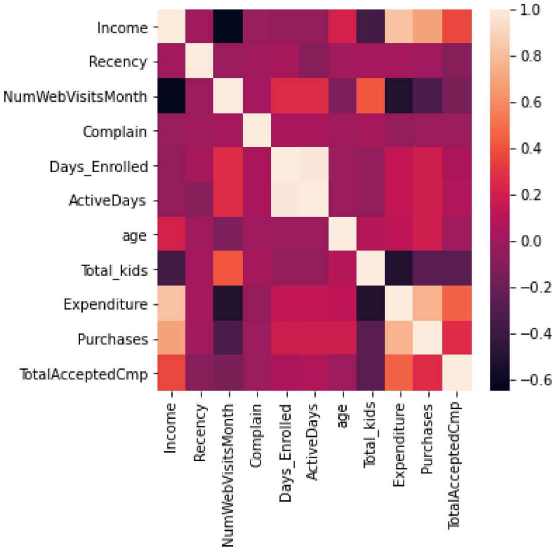
```
#There are some customers aged above 100. This is unlikely to happen. Let's drop those customers from data
#There are some customers who are earning more than 120,000 and some of them even more than 600,000. They are clearly the outliers i
df = df[df.age < 100]
df = df[df.Income < 120000]
```

In [22]:

```
#correlation matrix
corrmat= df.corr()
plt.figure(figsize=(5,5))
sns.heatmap(corrmat)
```

Out[22]:

```
<AxesSubplot:>
```



In [23]:

```
int_list = []
for col in df.columns:
    if df[col].dtypes == int or df[col].dtypes == float:
        int_list.append(col)
print(int_list)
```

```
['Income']
```

In [24]:

```
#Get list of categorical variables
s = (df.dtypes == 'object')
object_cols = list(s[s].index)

print("Categorical variables in the dataset:", object_cols)
```

```
Categorical variables in the dataset: ['Education', 'Marital_Status']
```

In [25]:

```
#Label Encoding the object dtypes.
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
for i in object_cols:
    df[i]=df[[i]].apply(LE.fit_transform)
```

In [26]:

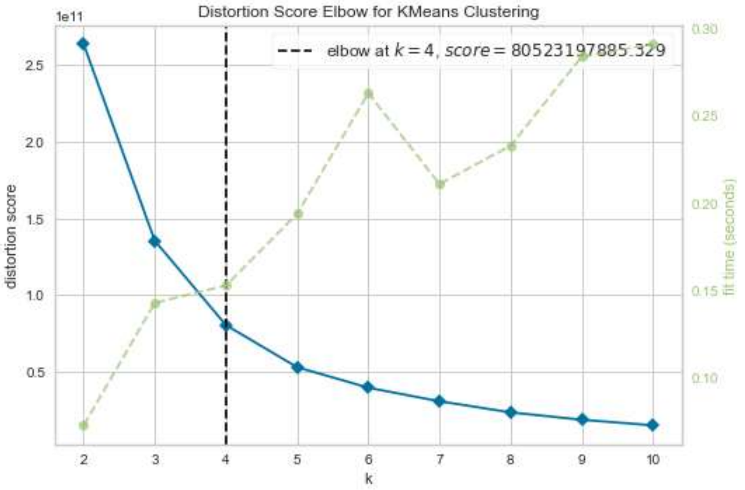
```
df.head()
```

Out[26]:

	Education	Marital_Status	Income	Recency	NumWebVisitsMonth	Complain	Days_Enrolled	ActiveDays	age	Total_kids	Expenditure	Purchases	TotalA
0	1	1	58138.0	58	7	0	971	913	57	0	1617	25	
1	1	1	46344.0	38	5	0	125	87	60	2	27	6	
2	1	0	71613.0	26	4	0	472	446	49	0	776	21	
3	1	0	26646.0	26	6	0	65	39	30	1	53	8	
4	0	0	58293.0	94	5	0	321	227	33	1	422	19	

In [27]:

```
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
model = KMeans(init = 'k-means++')
visualizer = KElbowVisualizer(model, k = 10, random_state = 42)
visualizer.fit(df)
visualizer.show()
```



Out[27]:

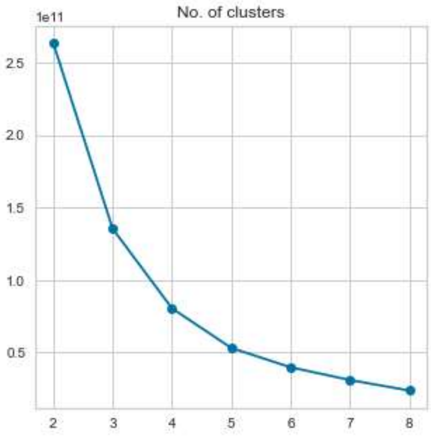
```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

In [28]:

```
from sklearn.cluster import KMeans

options = range(2,9)
inertias = []

for n_clusters in options:
    model = KMeans(n_clusters, random_state=42).fit(df)
    inertias.append(model.inertia_)

plt.figure(figsize=(5, 5))
plt.title("No. of clusters")
plt.plot(options, inertias, '-o')
```

Out[28]:

```
[<matplotlib.lines.Line2D at 0x228895a9d60>]
```



In [29]:

```
# Elbow got at k=4
kmeans = KMeans(n_clusters=4)
# Fit the algorithm to the features
kmeans.fit(df)

from sklearn.metrics import silhouette_score
# Compute the silhouette score
kmeans_silhouette = silhouette_score(
    df, kmeans.labels_
).round(2)
kmeans_silhouette
```

Out[29]:

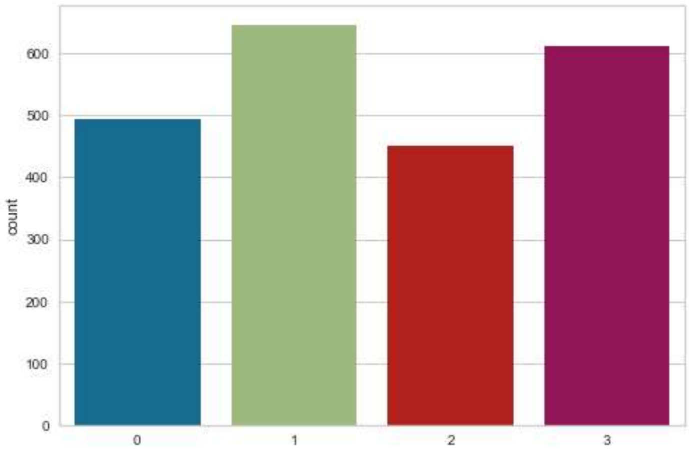
```
0.53
```

In [30]:

```
# countplot to check the number of clusters and number of customers in each cluster
y_clusters = kmeans.predict(df)
df['cluster_pca'] = y_clusters
sns.countplot(y_clusters)
```

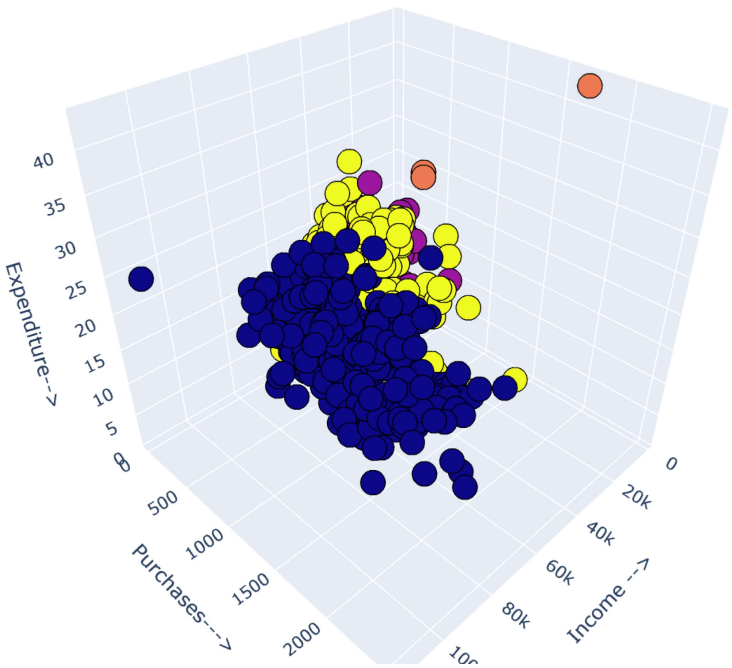
Out[30]:

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



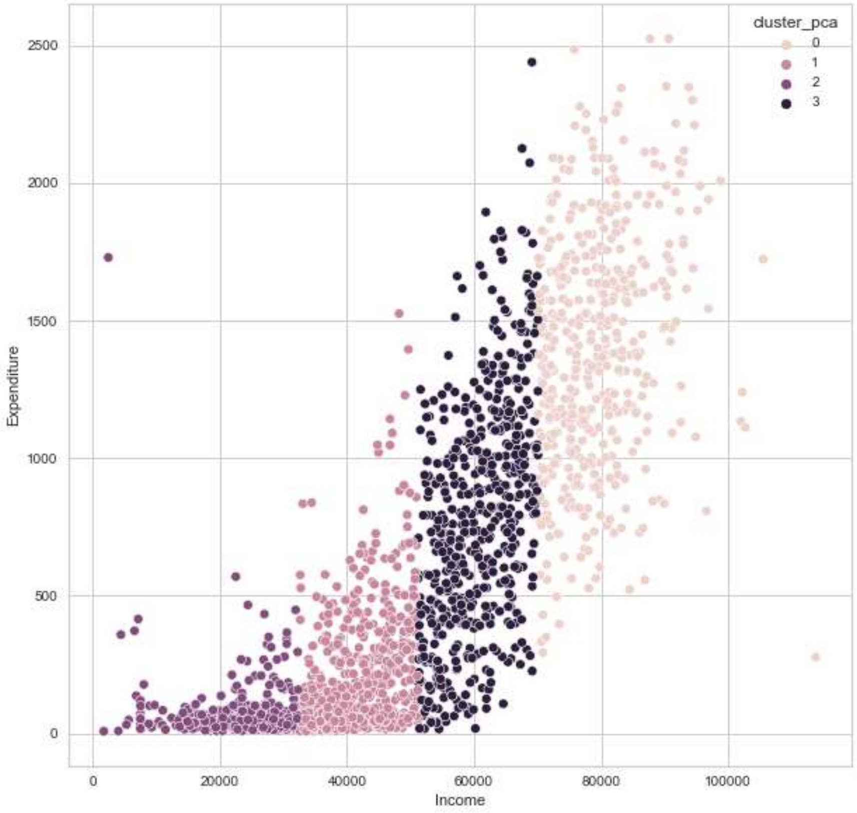
In [31]:

```
import matplotlib.pylab as plt # plotting
import plotly.graph_objs as go
import seaborn as sns

# 3d scatterplot using plotly
Scene = dict(xaxis = dict(title  = 'Income -->'),yaxis = dict(title  = 'Purchases--->'),zaxis = dict(title  = 'Expenditure-->'))

# model.labels_ is nothing but the predicted clusters i.e y_clusters
labels = kmeans.labels_
trace = go.Scatter3d(x=df['Income'], y=df['Expenditure'], z=df['Purchases'], mode='markers',marker=dict(color = labels, size= 10, li
layout = go.Layout(margin=dict(l=0,r=0),scene = Scene,height = 700,width = 700)
data = [trace]
fig = go.Figure(data = data, layout = layout)
fig.show()
```



In [32]:

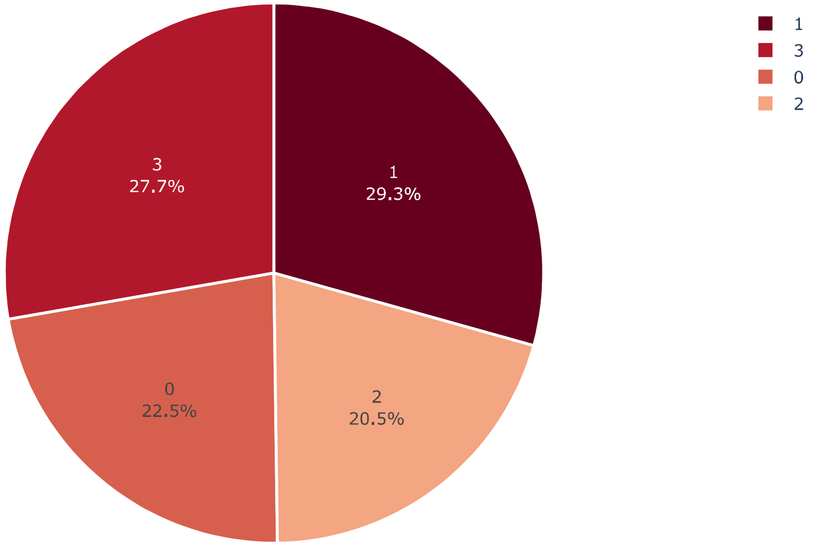
```
plt.figure(figsize=(10,10))
sns.scatterplot(data=df, x='Income', y='Expenditure', hue='cluster_pca');
plt.xlabel('Income')
plt.ylabel('Expenditure');
```



In [33]:

```
import plotly.express as px
cluster_counts = df.cluster_pca.value_counts()
```

```
fig = px.pie(cluster_counts,
             values = cluster_counts.values,
             names = cluster_counts.index,
             color_discrete_sequence=px.colors.sequential.RdBu)
fig.update_traces(textposition='inside', textinfo='percent+label',
                  marker = dict(line = dict(color = 'white', width = 2)))
fig.show()
```



```
In [ ]:
```