Executive Report

Customer Setgmentation Analysis

In [28]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN
```

In [2]:
```
data = pd.read_csv('./Customer_seg/customer_segmentation_data.csv')
data.head()
```

Out[2]:

	id	age	gender	income	spending_score	membership_years	purchase_frequency	preferred_category	last_purchase_amount
0	1	38	Female	99342	90	3	24	Groceries	113.53
1	2	21	Female	78852	60	2	42	Sports	41.93
2	3	60	Female	126573	30	2	28	Clothing	424.36
3	4	40	Other	47099	74	9	5	Home & Garden	991.93
4	5	65	Female	140621	21	3	25	Electronics	347.08

In [3]:
```
data.shape
```

Out[3]:
```
(1000, 9)
```

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
id                    0
age                   0
gender                0
income                0
spending_score        0
membership_years      0
purchase_frequency    0
preferred_category    0
last_purchase_amount  0
dtype: int64
```

In [5]:
```
data.describe()
```

Out[5]:

	id	age	income	spending_score	membership_years	purchase_frequency	last_purchase_amount
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000
mean	500.500000	43.783000	88500.800000	50.685000	5.46900	26.596000	492.348670
std	288.819436	15.042213	34230.771122	28.955175	2.85573	14.243654	295.744253
min	1.000000	18.000000	30004.000000	1.000000	1.00000	1.000000	10.400000
25%	250.750000	30.000000	57911.750000	26.000000	3.00000	15.000000	218.762500
50%	500.500000	45.000000	87845.500000	50.000000	5.00000	27.000000	491.595000
75%	750.250000	57.000000	116110.250000	76.000000	8.00000	39.000000	747.170000
max	1000.000000	69.000000	149973.000000	100.000000	10.00000	50.000000	999.740000

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    1000 non-null   int64
 1   age                   1000 non-null   int64
 2   gender                1000 non-null   object
 3   income                1000 non-null   int64
 4   spending_score        1000 non-null   int64
 5   membership_years      1000 non-null   int64
 6   purchase_frequency    1000 non-null   int64
 7   preferred_category    1000 non-null   object
 8   last_purchase_amount  1000 non-null   float64
dtypes: float64(1), int64(6), object(2)
memory usage: 70.4+ KB
```

In [8]:
```
le = LabelEncoder()
gender_label = le.fit_transform(data['gender'])
gender_label_maping = dict(zip(le.classes_, le.transform(le.classes_)))
gender_label_maping
```

Out[8]:
```
{'Female': np.int64(0), 'Male': np.int64(1), 'Other': np.int64(2)}
```

In [9]:
```
pref_cat_label = le.fit_transform(data['preferred_category'])
pref_cat_label_maping = dict(zip(le.classes_, le.transform(le.classes_)))
pref_cat_label_maping
```

Out[9]:
```
{'Clothing': np.int64(0),
 'Electronics': np.int64(1),
 'Groceries': np.int64(2),
 'Home & Garden': np.int64(3),
 'Sports': np.int64(4)}
```

In [10]:
```
data_mod = data.drop(['id', 'gender','preferred_category'], axis=1)
data_mod['gender'] = gender_label
data_mod['preferred_category'] = pref_cat_label
```

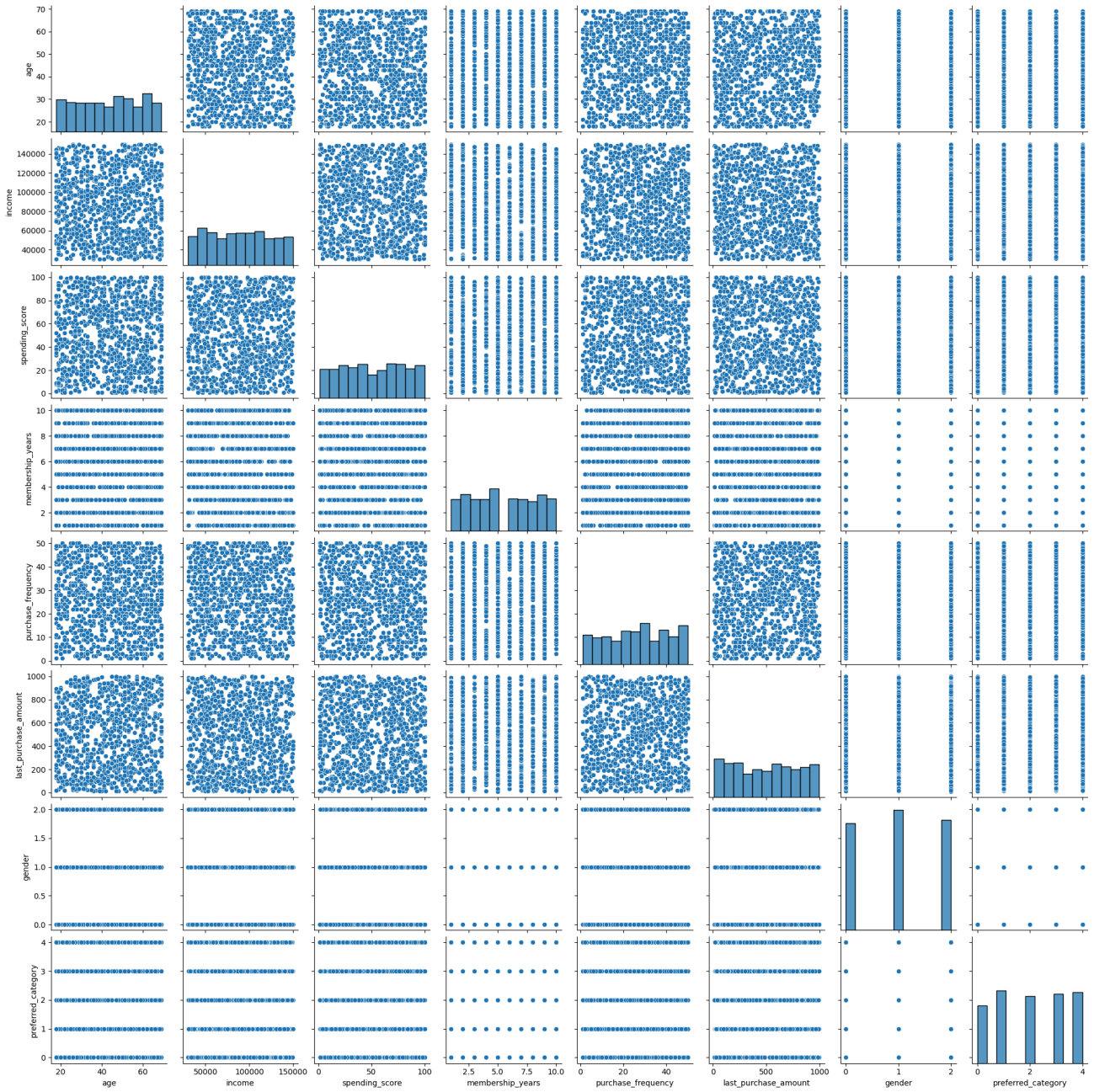
In [11]:
```
data_mod.describe()
```

Out[11]:

	age	income	spending_score	membership_years	purchase_frequency	last_purchase_amount	gender	preferred_category
count	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000
mean	43.783000	88500.800000	50.685000	5.46900	26.596000	492.348670	1.011000	2.071000
std	15.042213	34230.771122	28.955175	2.85573	14.243654	295.744253	0.802199	1.392084
min	18.000000	30004.000000	1.000000	1.00000	1.000000	10.400000	0.000000	0.000000
25%	30.000000	57911.750000	26.000000	3.00000	15.000000	218.762500	0.000000	1.000000
50%	45.000000	87845.500000	50.000000	5.00000	27.000000	491.595000	1.000000	2.000000
75%	57.000000	116110.250000	76.000000	8.00000	39.000000	747.170000	2.000000	3.000000
max	69.000000	149973.000000	100.000000	10.00000	50.000000	999.740000	2.000000	4.000000

In [12]:
```
sns.pairplot(data_mod)
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x722509a23340>
```



In [14]:
```
SS = StandardScaler()
X = SS.fit(data_mod).transform(data_mod)
```

In [15]:
```
pca2 = PCA(n_components=2)
X_PCA = pca2.fit(X).transform(X)
```

In [17]:
```
X_PCA
```

Out[17]:
```
array([[-1.88695207,  0.0291139 ],
       [-2.23483501,  0.82026241],
       [-0.95666988, -1.25667335],
       ...,
       [-1.52722355,  1.07236745],
       [-1.06486346,  1.65121928],
       [-0.78154912, -0.79105567]], shape=(1000, 2))
```

In [20]:
```
loadings_PCA = pd.DataFrame(
    pca2.components_.T,
    columns=['PCA1', 'PCA2'],
    index=data_mod.columns
)

print(loadings_PCA.sort_values(by='PCA1', ascending=False)['PCA1'])
```

```
last_purchase_amount    0.632869
gender                  0.541963
age                     0.388245
membership_years        0.207986
purchase_frequency      0.175795
spending_score          0.041662
preferred_category     -0.020734
income                 -0.280520
Name: PCA1, dtype: float64
```

Interpretación de PCA1

- Principales contribuyentes (coeficientes más altos en valor absoluto):

- last_purchase_amount (0.633) → mayor peso
- gender (0.542)
- age (0.388)

Entonces PCA1 mide principalmente:

Una combinación de comportamiento de compra y demografía, especialmente personas con alto monto de última compra, cierto género (codificado, por ejemplo 0/1), y mayor edad.

También hay influencia moderada de:

- membership_years y purchase_frequency (positivo)
- income (negativo, pero menor en magnitud)

Un valor alto en PCA1 → cliente con:

- Alta edad
- Alta última compra
- Probablemente de un género específico (según codificación)
- Más antigüedad como cliente

In [21]:
```
print(loadings_PCA.sort_values(by='PCA2', ascending=False)['PCA2'])
```

```
membership_years        0.647124
purchase_frequency      0.594039
spending_score          0.212101
preferred_category     -0.028038
gender                 -0.057221
income                 -0.141926
last_purchase_amount   -0.191902
age                    -0.349759
Name: PCA2, dtype: float64
```

Interpretación de PCA2

🔹 Principales contribuyentes:

- membership_years (0.647)
- purchase_frequency (0.594)
- spending_score (0.212)

Entonces PCA2 mide principalmente:

- Un componente de fidelidad o engagement, relacionado con el tiempo de membresía y frecuencia de compra.

Un valor alto en PCA2 → cliente activo y antiguo, con frecuente interacción o compras.

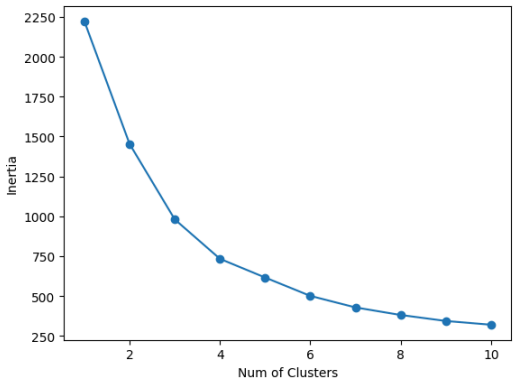
interpretar los clusters del GMM

- Si un cluster está en la parte alta de PCA1 y baja de PCA2, probablemente son clientes de alto poder adquisitivo o grandes compradores, pero no tan leales.
- Un cluster con alto PCA2 y bajo PCA1 puede ser de clientes fieles con menor nivel de gasto individual.
- Un cluster alto en ambos → clientes muy valiosos (grandes compras y alta frecuencia / antigüedad).
- Clusters bajos en ambos → clientes nuevos o poco comprometidos.

K-Means

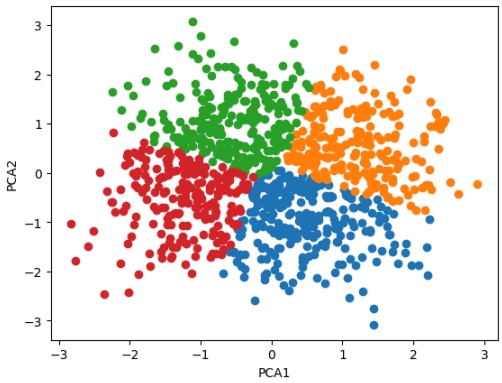
In [16]:
```
inertia = []
list_num_clusters = list(range(1,11))
for num_clusters in list_num_clusters:
    km = KMeans(n_clusters=num_clusters, random_state=42)
    km.fit(X_PCA)
    inertia.append(km.inertia_)
plt.plot(list_num_clusters, inertia)
plt.scatter(list_num_clusters, inertia)
plt.xlabel('Num of Clusters')
plt.ylabel('Inertia')
```

Out[16]:
```
Text(0, 0.5, 'Inertia')
```



In [24]:
```
km = KMeans(n_clusters=4, random_state=42)
km.fit(X_PCA)
labels = km.labels_
for label in np.unique(km.labels_):
    X_ = X_PCA[label == km.labels_]
    plt.scatter(X_[:,0], X_[:,1], label=label)

plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()
```



Gaussian Mixture

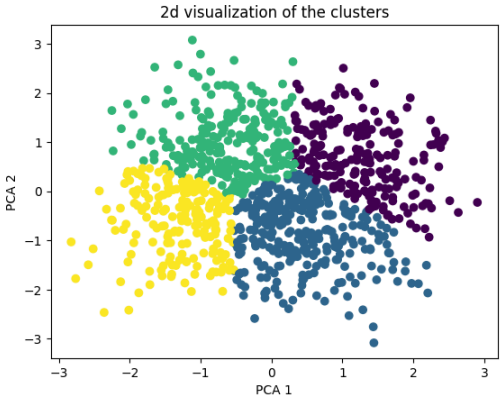
In [25]:
```
from sklearn.mixture import GaussianMixture
model = GaussianMixture(n_components=4, random_state=0)
model.fit(X_PCA)
```

Out[25]:
```
GaussianMixture(n_components=4, random_state=0)
```

In [26]:
```
X_pred = model.predict(X_PCA)
```

In [27]:
```
x = X_PCA[:,0]
y = X_PCA[:,1]
plt.scatter(x, y, c=X_pred)
plt.title("2d visualization of the clusters")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
```

Out[27]:
```
Text(0, 0.5, 'PCA 2')
```



Mean Shift Clustering

In [60]:
```
from sklearn.cluster import MeanShift, estimate_bandwidth
```

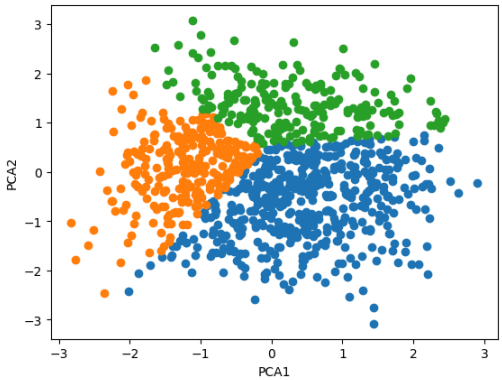
In [140…
```
bandwidth = estimate_bandwidth(X_PCA, quantile=.07, n_samples=1000)
bandwidth
```

Out[140…
```
np.float64(0.658670257728277)
```

In [141…
```
ms = MeanShift(bandwidth=bandwidth,bin_seeding=True)
ms.fit(X_PCA)
labeled=ms.labels_
clusters=ms.predict(X_PCA)
```

In [142…
```
labels = km.labels_
for label in np.unique(ms.labels_):
    X_ = X_PCA[label == ms.labels_]
    plt.scatter(X_[:,0], X_[:,1], label=label)

plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()
```



In [ ]: