Segmentos y Churn de Hotel VIP

Objetivo:

Como dueño de un hotel famoso y con muchas transacciones, registro comercial pendiente a confirmar, deseo saber cuales son mis segmentos de clientes y que segmento es el que tiene mayor perdida porcentual de clientes.

Datos:

- tlacuachitos_vip_transactions.csv
- tlacuachitos_vip_customers_data.csv

Actividad 1)

- Elabora y explica segmentos de clientes que podría usar para realizar mi dirección estratégica.

Actividad 2)

- Calcula y obtén el porcentaje de clientes perdidos que tengo en cada segmento

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans, DBSCAN
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import set_config
set_config(working_memory=1024)
```

Actividad 1

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

Out[2]:

	CustomerID	Age	Income	Tenure	Education	Industry	Geographic Location	Churn_Risk	Cohort
0	1	56	52752.677346	3	Master	Technology	Europe	1	2023-08-31
1	2	69	55297.364348	6	Bachelor	Technology	South America	0	2021-08-31
2	3	46	57978.753383	3	Bachelor	Finance	Europe	1	2019-05-31
3	4	32	60445.266900	3	High School	Education	South America	1	2021-02-28
4	5	60	57741.870929	5	Bachelor	Entertainment	Asia	0	2018-10-31

In [3]:
```
df_transactions = pd.read_csv('tlacuachitos_vip_transactions.csv')
df_transactions.head()
```

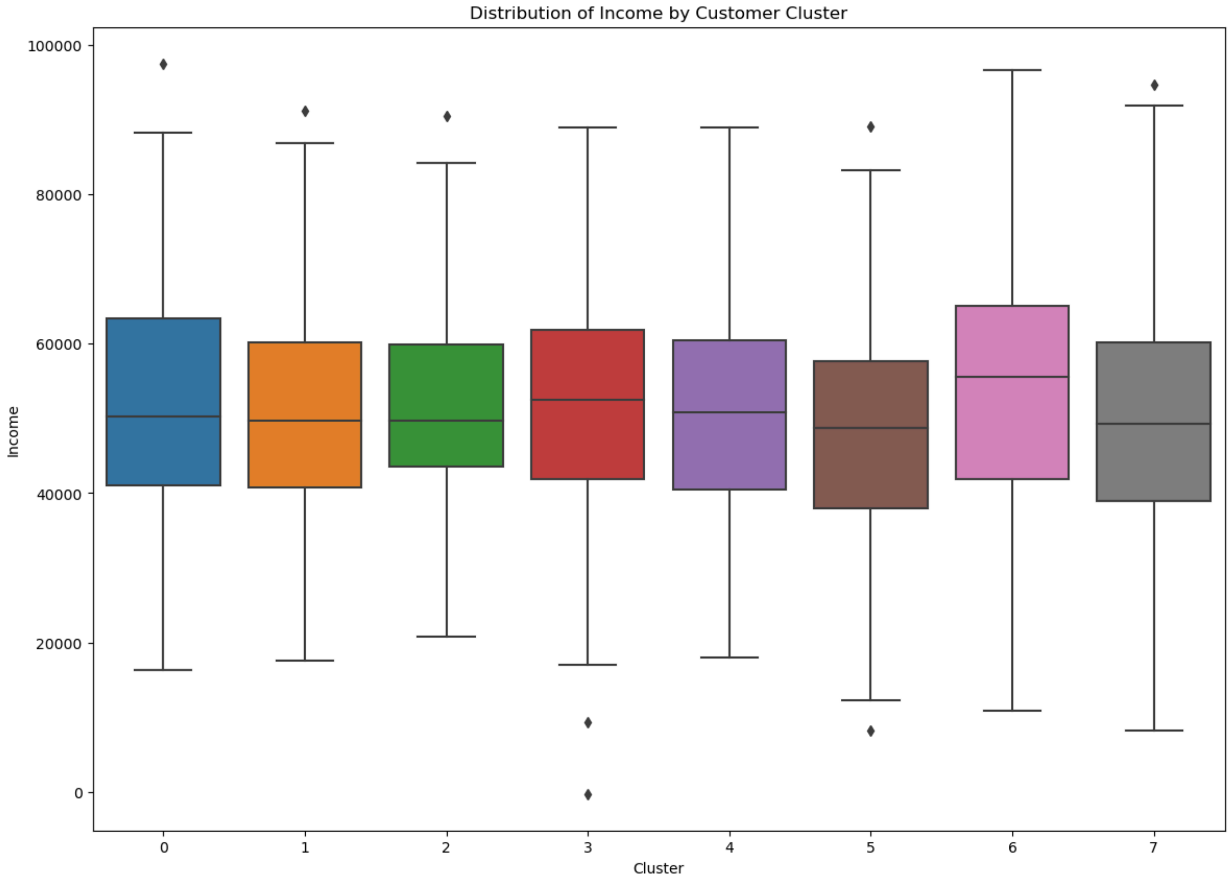
Out[3]:

	CustomerID	TransactionDate	TransactionAmount
0	1	2023-10-31	518.444092
1	1	2024-07-31	353.796197
2	1	2024-01-31	38.206591
3	1	2024-06-30	724.929423
4	2	2022-02-28	145.616000

In [4]:
```
# Convertir variables categóricas en variables dummies
categorical_features = ['Education','Industry','Geographic Location']
data_encoded = pd.get_dummies(data, columns=categorical_features, drop_first=True)

# Preparar los datos seleccionando características relevantes y normalizándolos
features = ['Age', 'Income'] + list(data_encoded.columns[7:])  # Incluir característic
x = data_encoded.loc[:, features].values
x = StandardScaler().fit_transform(x)  # Normalización de las características

# Aplicar K-means clustering para identificar segmentos de clientes
kmeans = KMeans(n_clusters=8, random_state=42)
labels = kmeans.fit_predict(x)

# Agregar las etiquetas del cluster al DataFrame original para análisis
data_encoded['Cluster'] = labels

# Visualizar los resultados del clustering
plt.figure(figsize=(14, 10))

# Create a boxplot to visualize the distribution of 'total_bill' for each 'Cluster'
sns.boxplot(x='Cluster', y='Income', data=data_encoded)

# Set labels and title
plt.xlabel('Cluster')
plt.ylabel('Income')
plt.title('Distribution of Income by Customer Cluster')
```

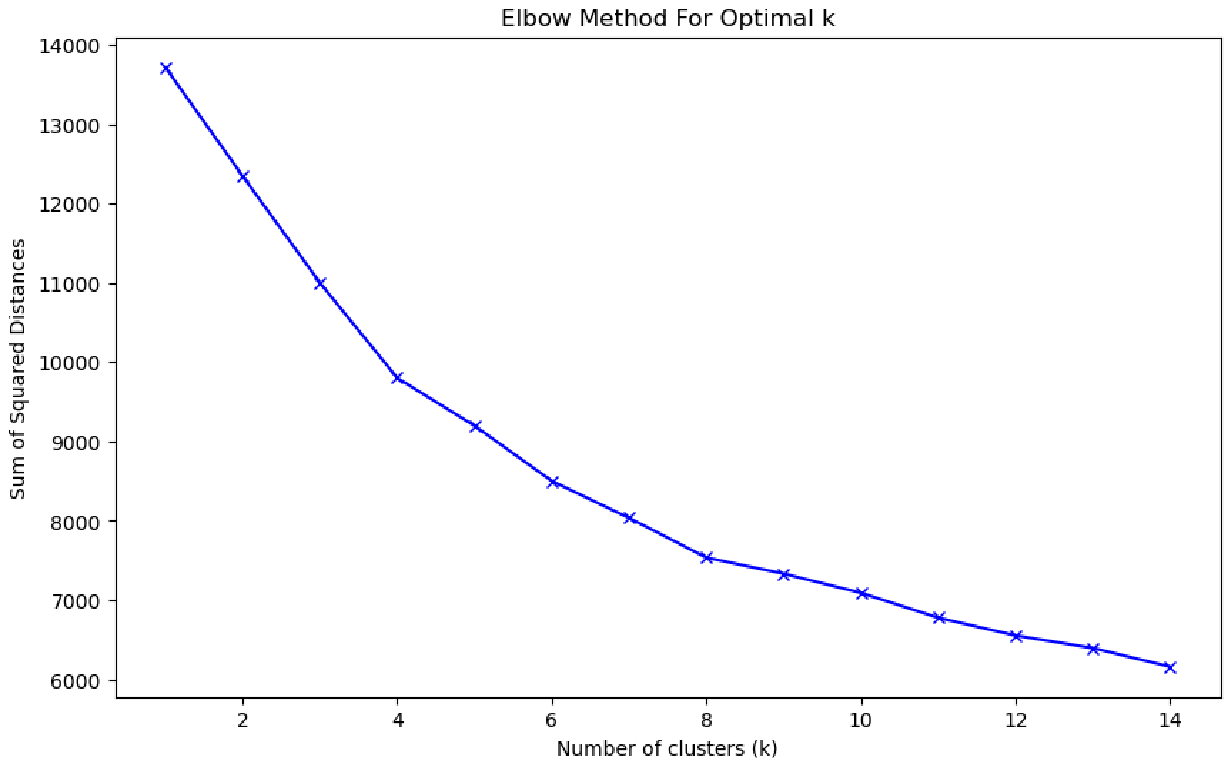
```
# Display the plot
plt.show()
```

```
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [5]:
```
# Determinar el número óptimo de clusters usando el método del codo
sum_of_squared_distances = []
K = range(1, 15)  # Ajuste el rango según sea necesario
for k in K:
    km = KMeans(n_clusters=k, random_state=42)
    km = km.fit(x)
    sum_of_squared_distances.append(km.inertia_)

# Plot the Elbow curve
plt.figure(figsize=(10, 6))
plt.plot(K, sum_of_squared_distances, 'bx-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Sum of Squared Distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

```
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



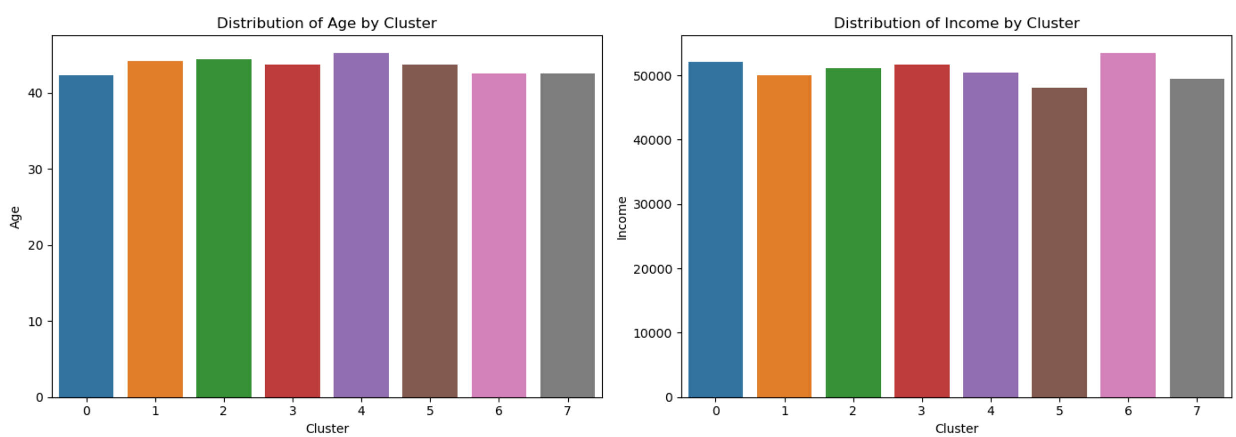
In [6]:
```
data_encoded.head()
```

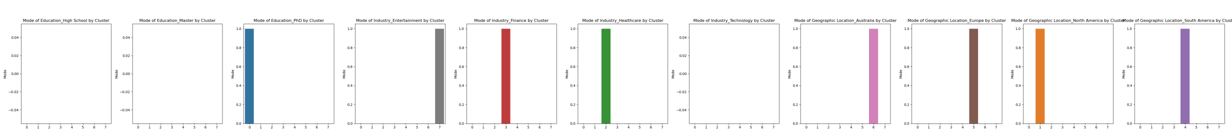
Out[6]:

	CustomerID	Age	Income	Tenure	Churn_Risk	Cohort	Education_High School	Education_Master	E
0	1	56	52752.677346	3	1	2023-08-31	0	1	
1	2	69	55297.364348	6	0	2021-08-31	0	0	
2	3	46	57978.753383	3	1	2019-05-31	0	0	
3	4	32	60445.266900	3	1	2021-02-28	1	0	
4	5	60	57741.870929	5	0	2018-10-31	0	0	

In [31]:
```
# Define a function to summarize the cluster characteristics
def summarize_clusters(data):
    # Numerical features
    numerical_features = ['Age','Income']  # Modify this list based on your dataset
    # Categorical features (assuming dummy variables were created)
    categorical_features = [col for col in data.columns if col.startswith('Education')

    # Summary DataFrame for numerical features
    cluster_summary_numerical = data.groupby('Cluster')[numerical_features].mean()

    # Summary DataFrame for categorical features
    cluster_summary_categorical = data.groupby('Cluster')[categorical_features].agg(la
```

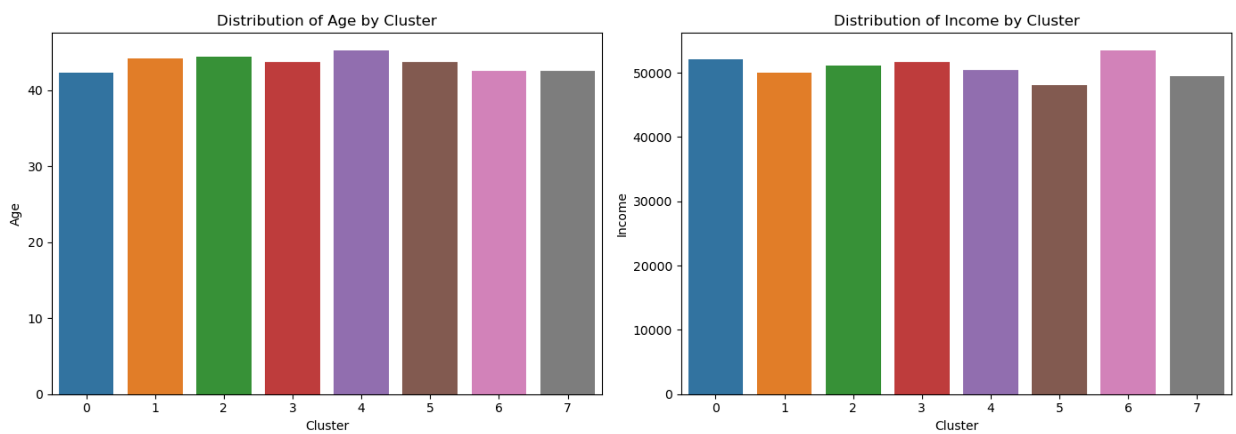
```
    return cluster_summary_numerical, cluster_summary_categorical

# Generate the summary for each cluster
cluster_summary_numerical, cluster_summary_categorical = summarize_clusters(data_encod

# Plotting
def plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical):
    # Plot numerical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_numerical.columns), fi
    for i, col in enumerate(cluster_summary_numerical.columns):
        sns.barplot(x=cluster_summary_numerical.index, y=col, data=cluster_summary_num
        axes[i].set_title(f'Distribution of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel(col)

    plt.tight_layout()
    plt.show()

    # Plot categorical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_categorical.columns),
    for i, col in enumerate(cluster_summary_categorical.columns):
        sns.barplot(x=cluster_summary_categorical.index, y=col, data=cluster_summary_c
        axes[i].set_title(f'Mode of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel('Mode')

    plt.tight_layout()
    plt.show()

# Call the plotting function
plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical)
```

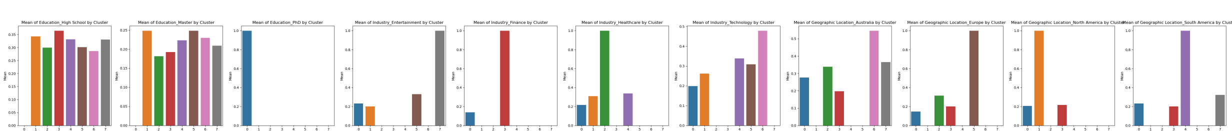




In [32]:
```
# Define a function to summarize the cluster characteristics
def summarize_clusters(data):
    # Numerical features
    numerical_features = ['Age','Income']  # Modify this list based on your dataset
    # Categorical features (assuming dummy variables were created)
    categorical_features = [col for col in data.columns if col.startswith('Education')

    # Summary DataFrame for numerical features
    cluster_summary_numerical = data.groupby('Cluster')[numerical_features].mean()
```

```
    # Summary DataFrame for categorical features
    cluster_summary_categorical = data.groupby('Cluster')[categorical_features].mean()

    return cluster_summary_numerical, cluster_summary_categorical

# Generate the summary for each cluster
cluster_summary_numerical, cluster_summary_categorical = summarize_clusters(data_encod

# Plotting
def plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical):
    # Plot numerical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_numerical.columns), fi
    for i, col in enumerate(cluster_summary_numerical.columns):
        sns.barplot(x=cluster_summary_numerical.index, y=col, data=cluster_summary_num
        axes[i].set_title(f'Distribution of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel(col)

    plt.tight_layout()
    plt.show()

    # Plot categorical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_categorical.columns),
    for i, col in enumerate(cluster_summary_categorical.columns):
        sns.barplot(x=cluster_summary_categorical.index, y=col, data=cluster_summary_c
        axes[i].set_title(f'Mean of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel('Mean')

    plt.tight_layout()
    plt.show()

# Call the plotting function
plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical)
```





Análisis de los clusters

El cluster 0 son solo de Phd, se reparten aproximadamente 20% en cada industria de entretenimiento, finanzas, salud y tecnología, además se reparten entre las difernetes regiones pero donde hay más es en asutralia pues son casi 30%. Por lo tanto se puede decir que el cluster 0 son personas con PHd en su mayoría en australia.

En el cluster 1 la mayoría esta en América del norte, un tercio tiene high school y un cuarto tiene maestría, se encuentran en tecnología, salud y entretenimiento, y todos son de américa del norte. Por lo que este cluster son las personas de américa del norte que no están en finanzas con mayoría trabajando en tecnología.

En el cluster 2 todos son del área de salud, el 30% tiene high school y el 25% maestría y un 30% está en australia y tan solo el 3% en Europa. Se puede decir que es el cluster de personas en área de salud mayoritariamente de australia.

En el cluster 3 todos son del área de finanzas repartidos por todo el mundo, ligeramente hay más en australia.

El cluster 4 un tercio tiene high school y un poco más del 20% tiene maetsría, varios trabajan en tecnología (+30%) y todos están en América del sur.

En el cluster 5 todos están en europa y poco más del 30% trabaja en tecnología y muy pocos en entretenimeinto.

En el cluster 6 como 25% tiene high school y 25% maestría, la mitad trabaja en tecnología, la mitad está en Australia, por lo que tienen mucha presencia en ese país.

En el cluster 7 todos estan en la industria de entretenimiento, un tercio tiene high school y un quinto tiene maestría, un tercio está en Australia y un cuarto en América del Sur.

El promedio de edad e ingreso por cluster es muy parecido para todos.

Actividad 2

In [9]:
```
df_transactions.head()
```

Out[9]:

	CustomerID	TransactionDate	TransactionAmount
0	1	2023-10-31	518.444092
1	1	2024-07-31	353.796197
2	1	2024-01-31	38.206591
3	1	2024-06-30	724.929423
4	2	2022-02-28	145.616000

In [10]:
```
data_encoded.head()
```

Out[10]:

	CustomerID	Age	Income	Tenure	Churn_Risk	Cohort	Education_High School	Education_Master	E
0	1	56	52752.677346	3	1	2023-08-31	0	1	
1	2	69	55297.364348	6	0	2021-08-31	0	0	
2	3	46	57978.753383	3	1	2019-05-31	0	0	
3	4	32	60445.266900	3	1	2021-02-28	1	0	
4	5	60	57741.870929	5	0	2018-10-31	0	0	

In [11]:
```
X = df_transactions.merge(data_encoded, on = 'CustomerID')
X.head()
```

Out[11]:

	CustomerID	TransactionDate	TransactionAmount	Age	Income	Tenure	Churn_Risk	Cohort
0	1	2023-10-31	518.444092	56	52752.677346	3	1	2023-08-31
1	1	2024-07-31	353.796197	56	52752.677346	3	1	2023-08-31
2	1	2024-01-31	38.206591	56	52752.677346	3	1	2023-08-31
3	1	2024-06-30	724.929423	56	52752.677346	3	1	2023-08-31
4	2	2022-02-28	145.616000	69	55297.364348	6	0	2021-08-31

In [12]:
```
X['TransactionDate'] = pd.to_datetime(X['TransactionDate'])
```

In [13]:
```
X.sort_values(by=['CustomerID', 'TransactionDate'])

X.head()
```

Out[13]:

	CustomerID	TransactionDate	TransactionAmount	Age	Income	Tenure	Churn_Risk	Cohort
0	1	2023-10-31	518.444092	56	52752.677346	3	1	2023-08-31
1	1	2024-07-31	353.796197	56	52752.677346	3	1	2023-08-31
2	1	2024-01-31	38.206591	56	52752.677346	3	1	2023-08-31
3	1	2024-06-30	724.929423	56	52752.677346	3	1	2023-08-31
4	2	2022-02-28	145.616000	69	55297.364348	6	0	2021-08-31

In [14]:
```
customer_invoices = X.groupby(['Cluster', 'CustomerID', 'TransactionDate'])['Transacti
customer_invoices.head()
```

Out[14]:

	Cluster	CustomerID	TransactionDate	TransactionAmount
0	0	25	2020-01-31	178.744747
1	0	34	2024-04-30	404.600578
2	0	35	2021-09-30	109.596265
3	0	35	2022-02-28	829.475474
4	0	35	2022-04-30	239.419359

In [15]:
```
sum(customer_invoices['TransactionAmount'] < 0)
```

Out[15]:
```
0
```

In [16]:
```
customer_invoices_clean = customer_invoices[customer_invoices['TransactionAmount'] > 0
```

Fechas

In [17]:
```
snapshot_date = customer_invoices_clean['TransactionDate'].max()
snapshot_date
```

Out[17]:
```
Timestamp('2024-08-31 00:00:00')
```

In [18]:
```
customer_invoices_clean['TransactionDate'].min()
```

Out[18]:
```
Timestamp('2018-01-31 00:00:00')
```

In [19]:
```
# Calculate days between purchases
customer_invoices_clean['DaysBetweenPurchases'] = customer_invoices_clean.groupby(['Cl
customer_invoices_clean.head()
```

Out[19]:

	Cluster	CustomerID	TransactionDate	TransactionAmount	DaysBetweenPurchases
0	0	25	2020-01-31	178.744747	NaN
1	0	34	2024-04-30	404.600578	NaN
2	0	35	2021-09-30	109.596265	NaN
3	0	35	2022-02-28	829.475474	151.0
4	0	35	2022-04-30	239.419359	61.0

In [20]:
```
general_threshold = customer_invoices_clean['DaysBetweenPurchases'].quantile(0.90)
general_threshold
```

Out[20]:
```
426.0
```

Tengo un poco más de 2100 días, por lo que 426 es una buena opción ya que no abarca tanto tiempo de nuestros datos.

In [21]:
```
threshold_per_country = customer_invoices_clean.groupby('Cluster')['DaysBetweenPurchas
    percentile_90=lambda x: x.quantile(0.90),
    transaction_count='count'
).reset_index()

threshold_per_country.head()
```

Out[21]:

	Cluster	percentile_90	transaction_count
0	0	394.6	295
1	1	396.0	422
2	2	421.2	332
3	3	455.3	478
4	4	395.0	320

In [22]:
```
threshold_per_country['threshold'] = threshold_per_country.apply(
    lambda row: general_threshold if row['transaction_count'] < 30 else row['percentil
    axis=1
)

threshold_per_country.head()
```

Out[22]:

	Cluster	percentile_90	transaction_count	threshold
0	0	394.6	295	394.6
1	1	396.0	422	396.0
2	2	421.2	332	421.2
3	3	455.3	478	455.3
4	4	395.0	320	395.0

In [23]:
```
last_invoice_date_per_customer = customer_invoices_clean.groupby(['Cluster','CustomerI

last_invoice_date_per_customer.head()
```

Out[23]:

	Cluster	CustomerID	LastTransactionDate
0	0	25	2020-01-31
1	0	34	2024-04-30
2	0	35	2023-03-31
3	0	39	2024-03-30
4	0	42	2024-06-30

In [24]:
```
customers = last_invoice_date_per_customer.merge(threshold_per_country, on='Cluster')
customers.head()
```

Out[24]:

	Cluster	CustomerID	LastTransactionDate	percentile_90	transaction_count	threshold
0	0	25	2020-01-31	394.6	295	394.6
1	0	34	2024-04-30	394.6	295	394.6
2	0	35	2023-03-31	394.6	295	394.6
3	0	39	2024-03-30	394.6	295	394.6
4	0	42	2024-06-30	394.6	295	394.6

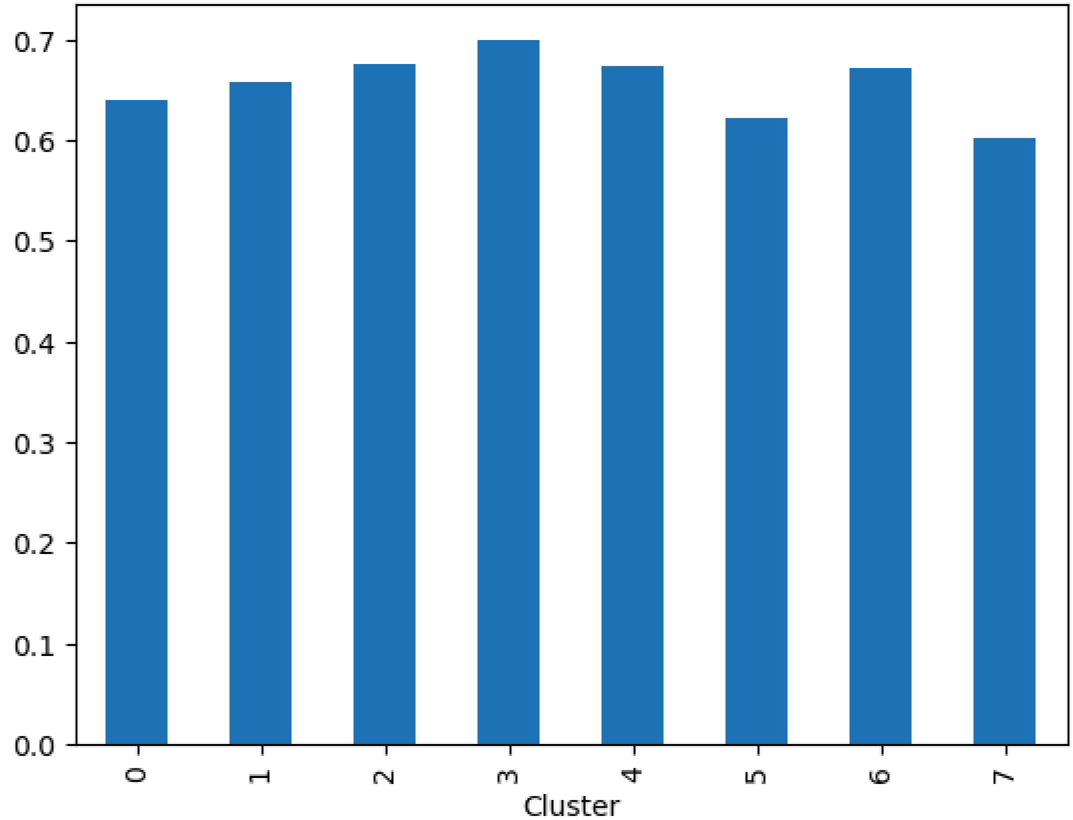
In [25]:
```
customers['inactivity_days'] = (snapshot_date - customers['LastTransactionDate']).dt.c
customers['churned'] = (customers['inactivity_days'] > customers['threshold']).astype(

customers.head()
```

Out[25]:

	Cluster	CustomerID	LastTransactionDate	percentile_90	transaction_count	threshold	inactivity_day
0	0	25	2020-01-31	394.6	295	394.6	167
1	0	34	2024-04-30	394.6	295	394.6	12
2	0	35	2023-03-31	394.6	295	394.6	51
3	0	39	2024-03-30	394.6	295	394.6	15
4	0	42	2024-06-30	394.6	295	394.6	6

In [26]:
```
customers.groupby('Cluster')['churned'].mean().plot(kind='bar')
```

Out[26]:
```
<Axes: xlabel='Cluster'>
```



La gráfica nos muestra que por segemnto se ha ido al menos un 60% de los clientes, esto no es buena señal para la empresa, pues hay clusters como el 3 que ha perdido 70% de los clientes. Esto nos dice que la empresa considera activos a un poco más del 30% de sus clientes como activos, esto es muy bajo para considerar que la empresa esta bien, más cuando tu mejor churn es del 65% que son muchos clientes, por lo que habrá que hacer mucho esfuerzo paar retener clientes y traer nuevos.