

```
In [ ]: #se importarán las siguientes librerías:  
import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
from sklearn.cluster import DBSCAN  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA  
from sklearn.metrics import silhouette_score  
from sklearn.model_selection import GridSearchCV  
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score
```

```
In [ ]: #Código para cargar el Dataset  
url = 'https://raw.githubusercontent.com/homerojc213/main/CONSOLIDADO_202112.csv'  
data=pd.read_csv(url, sep=";")  
dscopy=pd.read_csv(url, sep=";")
```

```
In [ ]: #Código que responde a la descripción anterior  
#Cantidad de Instancias y atributos
```

```
print('Cantidad de Instancias y columnas:', data.shape)  
print('Nombre columnas:', data.columns)
```

```
Cantidad de Instancias y columnas: (31829, 30)  
Nombre columnas: Index(['SK_CLIENTE', 'NO_PRODUCTO', 'SK_TRANSACCION', 'DS_NOMBRE',  
'SK_FE_TRANSACCION', 'SK_PRODUCTO_SERVICIO', 'DS_LINEA_PRODUCTO',  
'DS_FAMILIA', 'DS_CLASE', 'VALOR_MVTO', 'CANTIDAD_TRANSACCION',  
'DS_SECTOR_PIB', 'DS_TIPO_EMPRESA', 'DS_TIPO_PERSONA', 'DS_GENERO',  
'FE_VINCULACION_CLIENTE', 'DS_ESTADO_CIVIL', 'FE_NACIMIENTO',  
'DS_PAIS_NACIMIENTO', 'DS_NIVEL_ESTUDIOS', 'DS_PROFESION',  
'DS_OCUPACION', 'DS_TIPO_VIVIENDA', 'DS_RAZON_SOCIAL', 'ID_CLIENTE',  
'NO_PERSONAS_A_CARGO', 'REF_NUM', 'FE_APERTURA', 'DK_PERSONA', 'SK_RC'],  
dtype='object')
```

```
In [ ]: data.head()
```

	SK_CLIENTE	NO_PRODUCTO	SK_TRANSACCION	DS_NOMBRE	SK_FE_TRANSACCION	SK_PRODUCTO
0	2582480	110014145338	714	IVA SOBRE CHEQUERAS (IGUAL SER)	20211206	
1	2582480	110014145338	758	COBRO DE CHEQUERA (IGUAL SERIE)	20211206	
2	2582480	110014145338	644	ND.TIMBRES CHEQUERA	20211206	
3	2582480	110014145338	2562	CONSIGNACION AVAL	20211206	
4	2582480	110014145338	345	ABONOS POR A.C.H	20211206	

5 rows × 30 columns

In []:	data= data.drop(columns=['DS_FAMILIA', 'REF_NUM', 'DS_NOMBRE', 'DS_RAZON_SOCIAL'], errors='ignore') #data = data.replace('-', np.nan) print(data.shape) data.head()				
	(31829, 26)				
Out[]:	SK_CLIENTE NO_PRODUCTO SK_TRANSACCION SK_FE_TRANSACCION SK_PRODUCTO_SERVICIO DS_NOMBRE				
0	2582480	110014145338	714	20211206	11
1	2582480	110014145338	758	20211206	11
2	2582480	110014145338	644	20211206	11
3	2582480	110014145338	2562	20211206	11
4	2582480	110014145338	345	20211206	11

5 rows × 26 columns

```
In [ ]: missing_values= data.isnull().sum()  
total_cells = np.product(data.shape)  
total_missing= missing_values.sum()
```

```
# percent of data that is missing  
((total_missing/total_cells) * 100)
```

```
Out[ ]: 0.23200902901804593
```

```
In [ ]: total = data.isnull().sum().sort_values(ascending=False)  
percent_1 = data.isnull().sum()/data.isnull().count()*100  
percent_2 = (round(percent_1, 4)).sort_values(ascending=False)  
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])  
missing_data
```

	Total	%
FE_NACIMIENTO	1914	6.0134
FE_VINCULACION_CLIENTE	6	0.0189
NO_PRODUCTO	0	0.0000
DK_PERSONA	0	0.0000
FE_APERTURA	0	0.0000
NO_PERSONAS_A_CARGO	0	0.0000
ID_CLIENTE	0	0.0000
DS_TIPO_VIVIENDA	0	0.0000
DS_OCUPACION	0	0.0000
DS_PROFESION	0	0.0000
DS_NIVEL_ESTUDIOS	0	0.0000
DS_PAIS_NACIMIENTO	0	0.0000
DS_ESTADO_CIVIL	0	0.0000
SK_CLIENTE	0	0.0000
DS_GENERO	0	0.0000
DS_TIPO_PERSONA	0	0.0000
DS_TIPO_EMPRESA	0	0.0000
DS_SECTOR_PIB	0	0.0000
CANTIDAD_TRANSACCION	0	0.0000
VALOR_MVTO	0	0.0000
DS_CLASE	0	0.0000
DS_LINEA_PRODUCTO	0	0.0000
SK_PRODUCTO_SERVICIO	0	0.0000
SK_FE_TRANSACCION	0	0.0000
SK_TRANSACCION	0	0.0000
SK_RC	0	0.0000

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

```
Out[ ]: SK_CLIENTE          0  
NO_PRODUCTO        0  
SK_TRANSACCION      0  
SK_FE_TRANSACCION   0  
SK_PRODUCTO_SERVICIO 0  
DS_LINEA_PRODUCTO    0  
DS_CLASE            0  
VALOR_MVTO          0  
CANTIDAD_TRANSACCION 0  
DS_SECTOR_PIB        0  
DS_TIPO_EMPRESA      0  
DS_TIPO_PERSONA       0  
DS_GENERO           0  
FE_VINCULACION_CLIENTE 6  
DS_ESTADO_CIVIL      0  
FE_NACIMIENTO        1914  
DS_PAIS_NACIMIENTO    0  
DS_NIVEL_ESTUDIOS     0  
DS_PROFESION          0  
DS_OCUPACION          0  
DS_TIPO_VIVIENDA      0  
ID_CLIENTE           0  
NO_PERSONAS_A_CARGO    0  
FE_APERTURA          0  
DK_PERSONA            0  
SK_RC                0  
dtype: int64
```

```
In [ ]: data['FE_VINCULACION_CLIENTE'].fillna('FE_APERTURA', inplace= True)
```

```
In [ ]: data=data.dropna()
```

```
In [ ]: data=data.drop_duplicates()
```

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

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 29915 entries, 12 to 31828
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   SK_CLIENTE        29915 non-null   int64  
 1   NO_PRODUCTO       29915 non-null   int64  
 2   SK_TRANSACCION    29915 non-null   int64  
 3   SK_FE_TRANSACCION 29915 non-null   int64  
 4   SK_PRODUCTO_SERVICIO 29915 non-null   int64  
 5   DS_LINEA_PRODUCTO 29915 non-null   object  
 6   DS_CLASE          29915 non-null   object  
 7   VALOR_MVTO         29915 non-null   float64 
 8   CANTIDAD_TRANSACCION 29915 non-null   int64  
 9   DS_SECTOR_PIB      29915 non-null   object  
 10  DS_TIPO_EMPRESA    29915 non-null   object  
 11  DS_TIPO_PERSONA    29915 non-null   object  
 12  DS_GENERO          29915 non-null   object  
 13  FE_VINCULACION_CLIENTE 29915 non-null   object  
 14  DS_ESTADO_CIVIL    29915 non-null   object  
 15  FE_NACIMIENTO      29915 non-null   object  
 16  DS_PAIS_NACIMIENTO 29915 non-null   object  
 17  DS_NIVEL_ESTUDIOS   29915 non-null   object  
 18  DS_PROFESION        29915 non-null   object  
 19  DS_OCUPACION        29915 non-null   object  
 20  DS_TIPO_VIVIENDA    29915 non-null   object  
 21  ID_CLIENTE          29915 non-null   int64  
 22  NO_PERSONAS_A_CARGO 29915 non-null   int64  
 23  FE_APERTURA         29915 non-null   object  
 24  DK_PERSONA          29915 non-null   object  
 25  SK_RC               29915 non-null   int64  
dtypes: float64(1), int64(9), object(16)
memory usage: 6.2+ MB

```

```
In [ ]: data['FE_NACIMIENTO']= pd.to_datetime(data.FE_NACIMIENTO, format='%Y-%m-%d %H:%M:%S.%f')
data['FE_VINCULACION_CLIENTE']= pd.to_datetime(data.FE_VINCULACION_CLIENTE, format='%Y-%m-%d %H:%M:%S.%f')
data['FE_APERTURA']= pd.to_datetime(data.FE_APERTURA, format='%Y-%m-%d %H:%M:%S.%f', errors='coerce')
data['SK_FE_TRANSACCION']= pd.to_datetime(data.SK_FE_TRANSACCION, format='%Y%m%d', errors='coerce')
```

```
In [ ]: #Se agregan 3 calculos para La Edad, antiguedad del cliente y tiempo de apertura del producto
today = pd.datetime.now()
data["EDAD_ANIOS"] = np.floor(((today - data['FE_NACIMIENTO']).dt.days)/365)
data["ANTIGUEDAD_ANIOS"] = np.floor(((today - data['FE_VINCULACION_CLIENTE']).dt.days))
data["APERTURA_ANIOS"] = np.floor(((today - data['FE_APERTURA']).dt.days)/365)

# Se extrae mes de La transaccion
data["Mes_Transact"] = pd.DatetimeIndex(data["SK_FE_TRANSACCION"]).month
```

```
<ipython-input-14-5a785aa58637>:3: FutureWarning: The pandas.datetime class is deprecated and will be removed from pandas in a future version. Import from datetime module instead.
```

```
today = pd.datetime.now()
```

```
In [ ]: #Cantidad de Clientes --> 17464
unique_customers = set(data.SK_CLIENTE.unique())
print("Numero de Clientes: ", len(unique_customers))
```

Numero de Clientes: 17465

In []: #Porcentaje de Distribucion por Linea de Producto

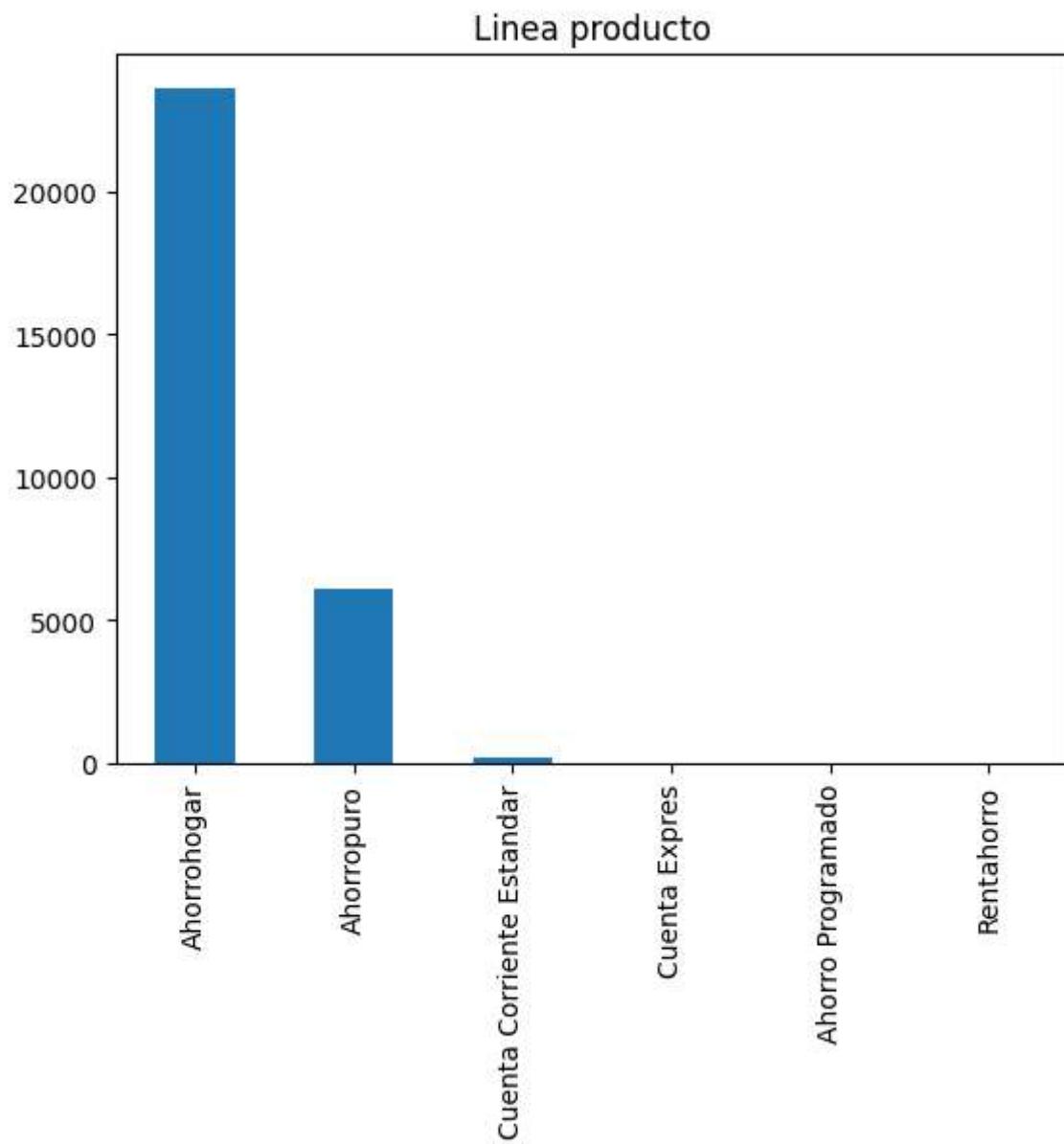
```
100*data['DS_LINEA_PRODUCTO'].value_counts()/len(data['DS_LINEA_PRODUCTO'])
```

Out[]:

Ahorrohogar	78.863446
Ahorropuro	20.300852
Cuenta Corriente Estandar	0.708675
Cuenta Expres	0.113655
Ahorro Programado	0.006686
Rentahorro	0.006686
Name: DS_LINEA_PRODUCTO, dtype: float64	

In []: #Gráfico de barras tendencia Linea Producto

```
plot=data['DS_LINEA_PRODUCTO'].value_counts().plot(kind="bar", title="Linea producto")
```



In []:

"""

Caracterizacion de los clientes:

* El 14 % de las transacciones estan asociadas a clientes entre los 68 y 75 años d

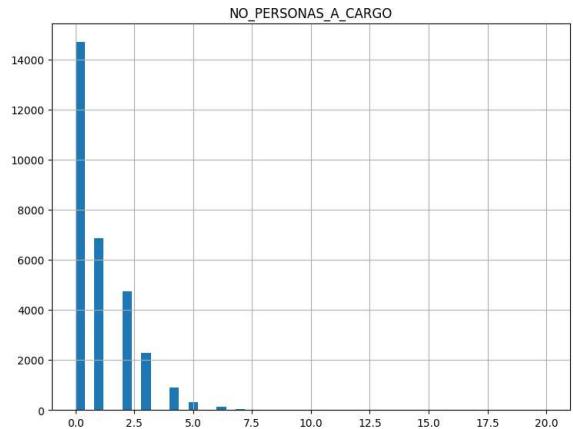
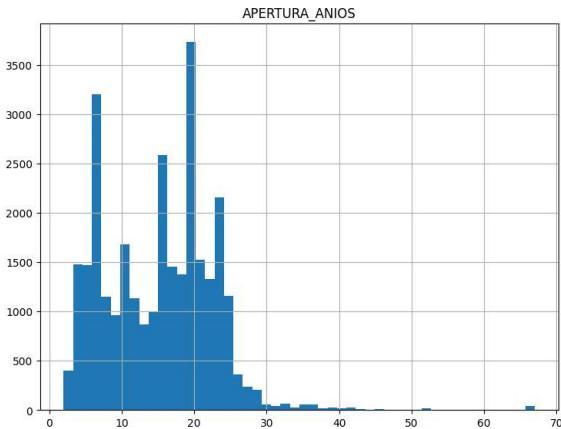
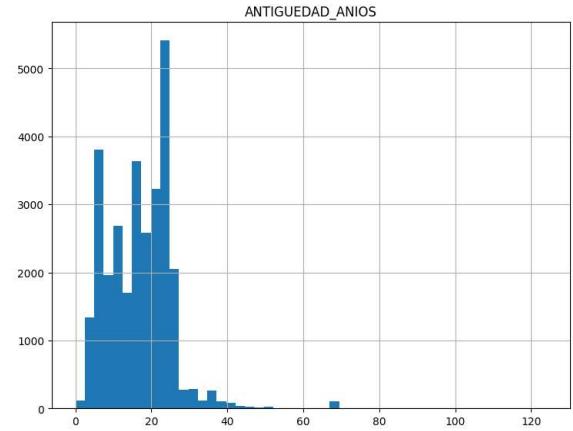
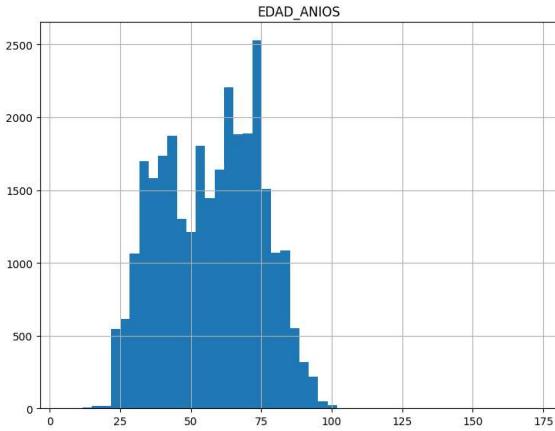
```

    * El 14 % de las transacciones estan asignadas a clientes con una antiguedad de 24
    * El 22% de las trasacciones estan dados sobre productos con 4-5-6 y 7 años de ape
    con 20 años de apertura
    * El 50% de los clientes del análisis no tienen personas a cargo, hay un 36 % rest
"""

data_pre=data[['EDAD_ANIOS', 'ANTIGUEDAD_ANIOS', 'APERTURA_ANIOS', 'NO_PERSONAS_A_CARGO']]

data_pre.hist(bins=50, figsize=(20,15))
plt.show()

```



```

In [ ]: # Distribucion de Las Edades

fig = plt.figure(figsize = (16,5))

#distplot
#ax1 = fig.add_subplot(121)
sns.distplot(data["EDAD_ANIOS"], kde=True)

#Cajas
ax1 = fig.add_subplot(122)
sns.distplot(data["ANTIGUEDAD_ANIOS"], kde=True)

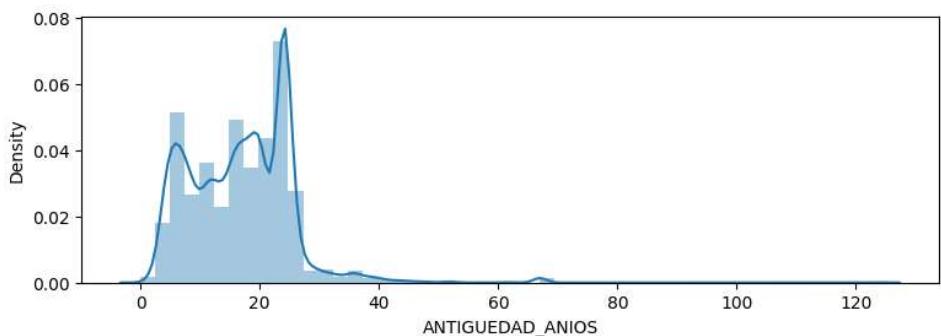
# Mostrar figura
fig.suptitle("Distribucion De la Edad y Antiguedad", fontsize=20)
plt.tight_layout(pad=5, w_pad=0.5, h_pad=.1)
plt.show()

```

```
<ipython-input-19-adcc706cadd8>:12: UserWarning:  
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.  
Please adapt your code to use either `displot` (a figure-level function with  
similar flexibility) or `histplot` (an axes-level function for histograms).  
For a guide to updating your code to use the new functions, please see  
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(data["ANTIGUEDAD_ANIOS"], kde=True)
```

Distribucion De la Edad y Antiguedad



In []:

```
"""  
Visualizacion valores por variable descriptiva  
    * La clase de productos analizados son Cuentas de Ahorro (99%) y Cuentas Corriente  
    * Los propietarios de los productos son en un 98% de Colombia y el restante de 48  
"""  
  
print("\nValores: Variable DS_LINEA_PRODUCTO: \n\n",data['DS_LINEA_PRODUCTO'].value_counts())  
print("\nValores: Variable DS_CLASE: \n\n",data['DS_CLASE'].value_counts())  
print("\nValores: Variable DS_PAIS_NACIMIENTO: \n\n",data['DS_PAIS_NACIMIENTO'].value_counts())  
print("\nValores: Variable DS_NIVEL_ESTUDIOS: \n\n",data['DS_NIVEL_ESTUDIOS'].value_counts())  
print("\nValores: Variable DS_OCUPACION: \n\n",data['DS_OCUPACION'].value_counts())  
print("\nValores: Variable DS_ESTADO_CIVIL: \n\n",data['DS_ESTADO_CIVIL'].value_counts())
```

Valores: Variable DS_LINEA_PRODUCTO:

Ahorrohogar	23592
Ahorropuro	6073
Cuenta Corriente Estandar	212
Cuenta Expres	34
Ahorro Programado	2
Rentahorro	2

Name: DS_LINEA_PRODUCTO, dtype: int64

Valores: Variable DS_CLASE:

Cuentas de Ahorro	29703
Cuentas Corrientes	212

Name: DS_CLASE, dtype: int64

Valores: Variable DS_PAIS_NACIMIENTO:

Colombia	29876
Venezuela	9
Ecuador	6
Italia	5
Espa?a	4
Chile	3
Holanda	2
Polonia	2
Peru	2
-	2
Islas Cook	1
Gran Bret?a e Irlanda del Norte	1
Brasil	1
Alemania	1

Name: DS_PAIS_NACIMIENTO, dtype: int64

Valores: Variable DS_NIVEL_ESTUDIOS:

Primaria	11904
Profesional	8933
Tecnico Profesional	3996
Especializacion	3431
Sin estudio	1561
-	48
Tecnologico	20
Maestria	14
Doctorado	4
Secundaria	4

Name: DS_NIVEL_ESTUDIOS, dtype: int64

Valores: Variable DS_OCUPACION:

Pensionado	14667
Policia Nacional	4884
Empleado Privado	4682
Docente	2902
Empleado Publico	851
Fuerzas Militares	667
Independiente	604
Estudiante	490
Hogar	168

Name: DS_OCUPACION, dtype: int64

Valores: Variable DS_ESTADO_CIVIL:

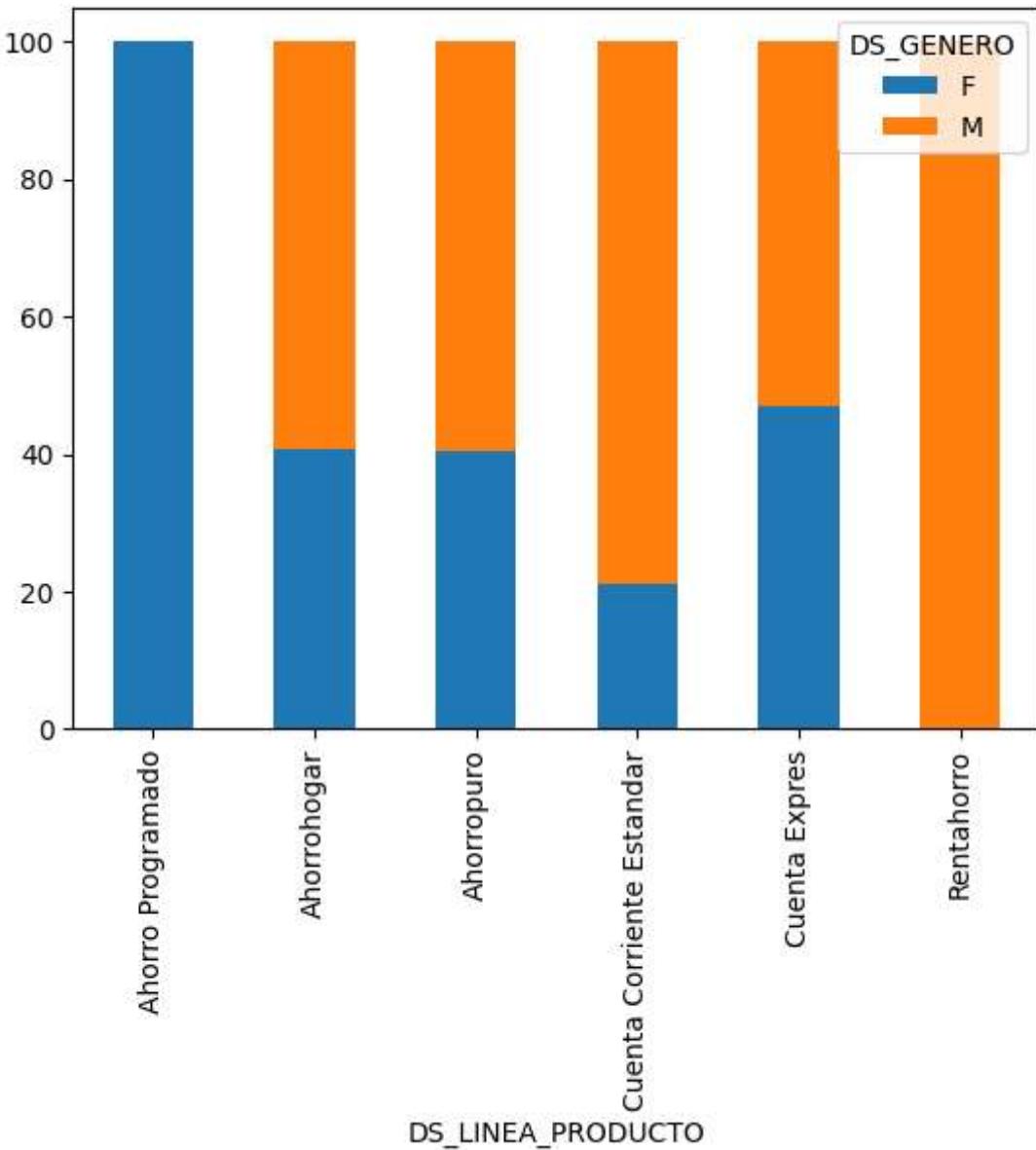
```
Casado(a)      11413
Soltero(a)     10118
Union Libre    3333
Viudo(a)       2599
Desconocido    1035
Separado(a)    969
Divorciado(a)  445
-              3
Name: DS_ESTADO_CIVIL, dtype: int64
```

```
In [ ]: """Las siguientes tablas de contingencia permiten observar la distribucion de algunas con respecto a la Linea de producto"""
"""

# Tabla de contingencia Linea Producto / Genero (Porcentaje)

pd.crosstab(index=data[ 'DS_LINEA_PRODUCTO' ],
             columns=data[ 'DS_GENERO' ]). apply(lambda r: r/r.sum() *100, axis=1).plot(kw['c'])

Out[ ]: <Axes: xlabel='DS_LINEA_PRODUCTO'>
```

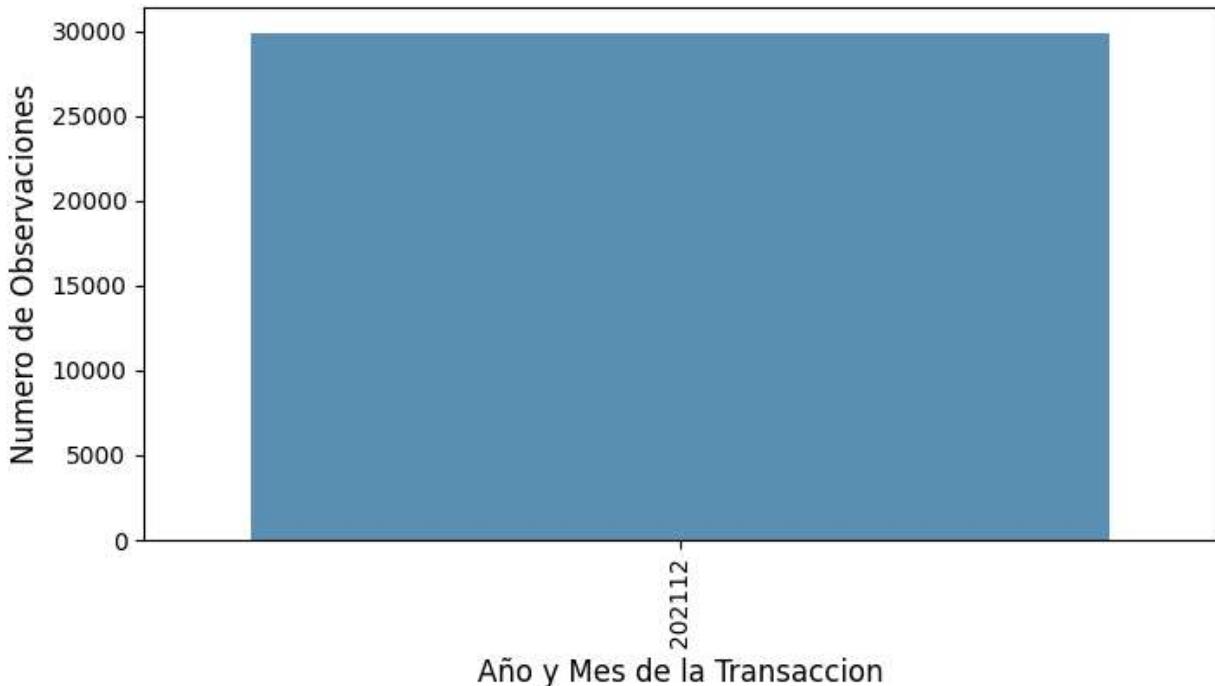


```
In [ ]: # Transacciones por mes

import seaborn as sns

data['fecha_dato_yearmonth'] = data['SK_FE_TRANSACCION'].apply(lambda x: (100*x.year)
yearmonth = data['fecha_dato_yearmonth'].value_counts()

plt.figure(figsize=(8,4))
sns.barplot(x=yearmonth.index, y=yearmonth.values, alpha=0.8)
plt.xlabel('Año y Mes de la Transaccion', fontsize=12)
plt.ylabel('Numero de Observaciones', fontsize=12)
plt.xticks(rotation='vertical')
plt.show()
```



```
In [ ]: #Funcion para definir categoria segun La Edad
```

```
def age_cat(x):
    if int(x) < 24:
        return('Youth')
    elif int(x) < 55:
        return('adults')
    else:
        return('old')

data=data.dropna()
data['Edad_cat']=data['EDAD_ANIOS'].apply(lambda x:age_cat(x))
```

```
<ipython-input-25-531c72fe9986>:12: 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\_guide/indexing.html#returning-a-view-versus-a-copy
data['Edad_cat']=data['EDAD_ANIOS'].apply(lambda x:age_cat(x))
```

```
In [ ]: #Visualizacion categoria por Edad
```

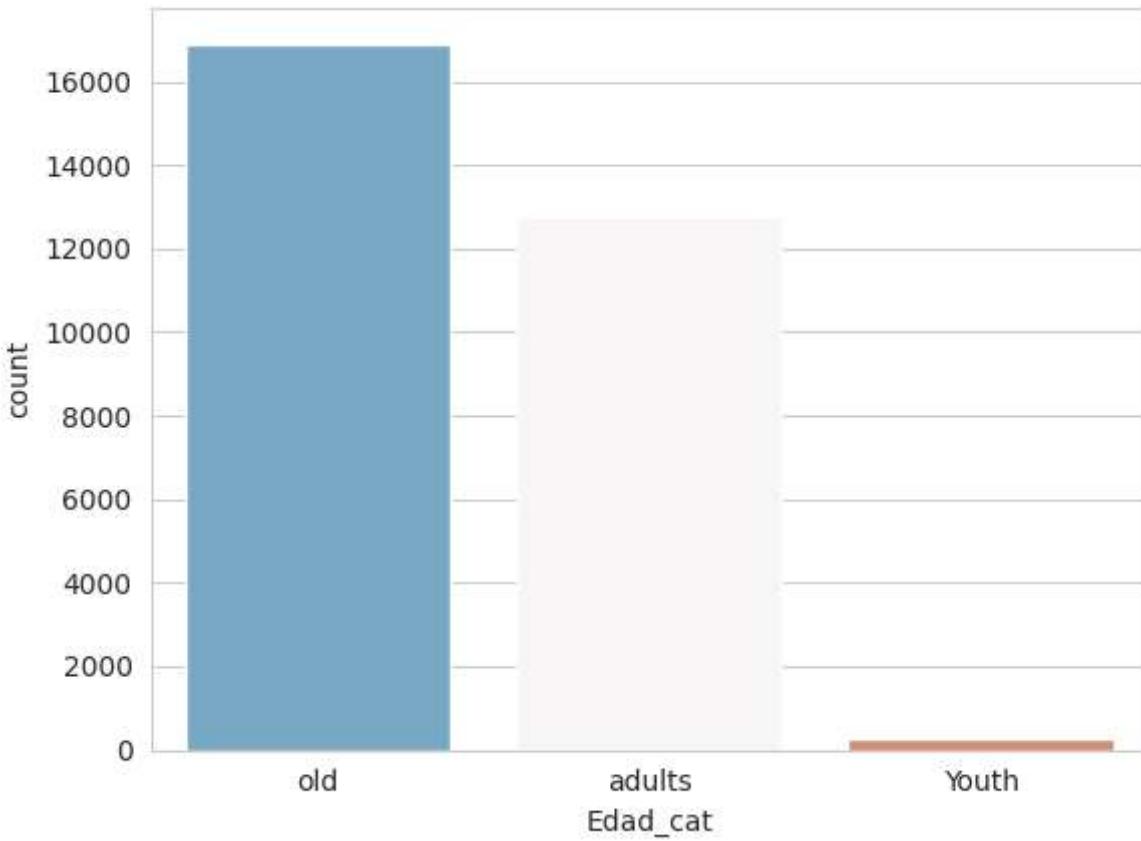
```
sns.set_style('whitegrid')
sns.countplot(x='Edad_cat',data=data,palette='RdBu_r')
```

```
<ipython-input-26-dd706990cbdc>:4: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.countplot(x='Edad_cat',data=data,palette='RdBu_r')
```

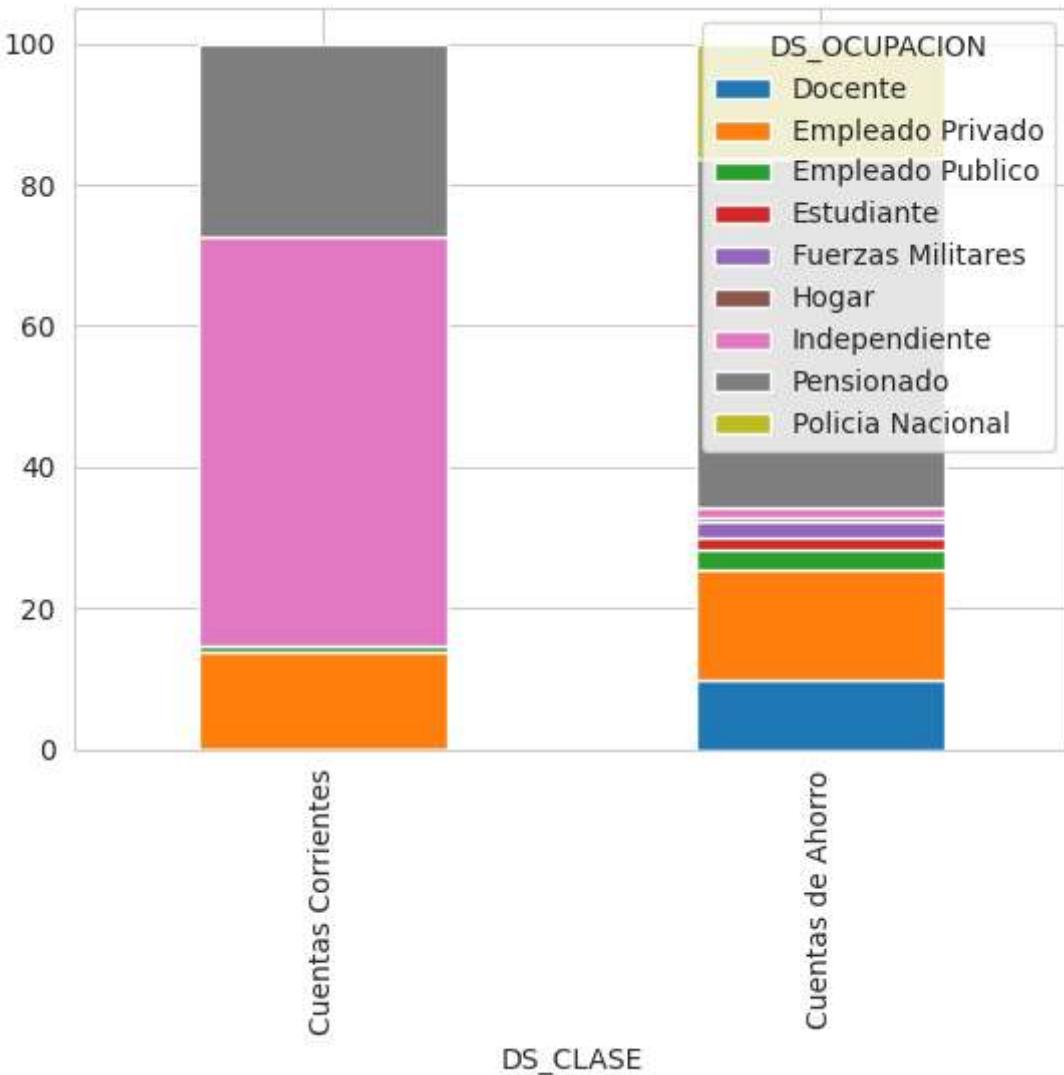
```
Out[ ]:
```



```
In [ ]: # Visualizacion DS_CLASE / DS_OCUPACION (Porcentaje)
```

```
pd.crosstab(index=data['DS_CLASE'],
             columns=data['DS_OCUPACION']).apply(lambda r: r/r.sum() *100, axis=1).plot
```

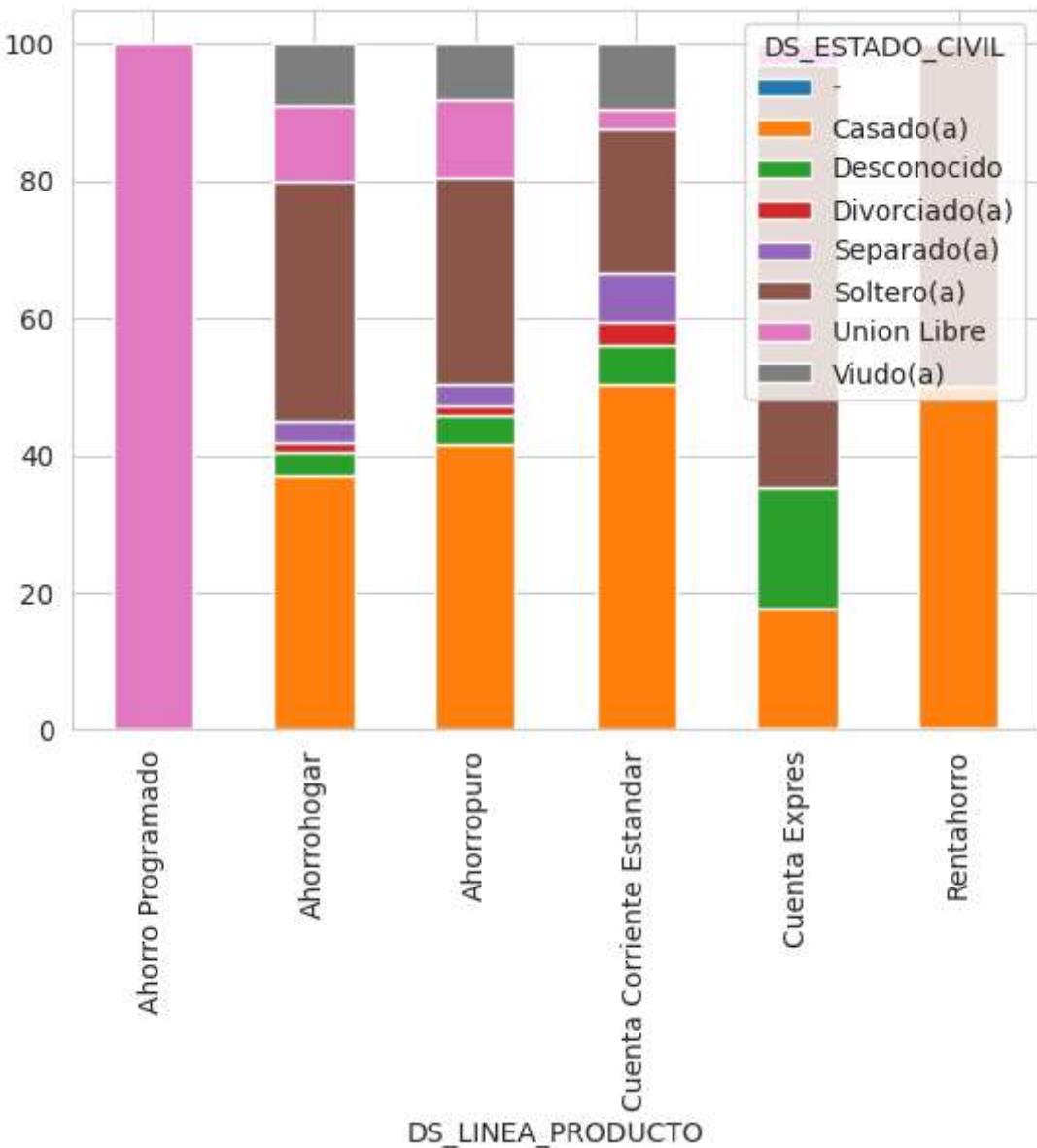
```
Out[ ]: <Axes: xlabel='DS_CLASE'>
```



```
In [ ]: # Visualizacion DS_LINEA_PRODUCTO / DS_ESTADO_CIVIL (Porcentaje)
```

```
pd.crosstab(index=data['DS_LINEA_PRODUCTO'],
             columns=data['DS_ESTADO_CIVIL']).apply(lambda r: r/r.sum() *100, axis=1).
```

```
Out[ ]: <Axes: xlabel='DS_LINEA_PRODUCTO'>
```



```
In [ ]: #identificamos las nuevas variables
```

```
data.columns
```

```
Out[ ]: Index(['SK_CLIENTE', 'NO_PRODUCTO', 'SK_TRANSACCION', 'SK_FE_TRANSACCION',
   'SK_PRODUCTO_SERVICIO', 'DS_LINEA_PRODUCTO', 'DS_CLASE', 'VALOR_MVTO',
   'CANTIDAD_TRANSACCION', 'DS_SECTOR_PIB', 'DS_TIPO_EMPRESA',
   'DS_TIPO_PERSONA', 'DS_GENERO', 'FE_VINCULACION_CLIENTE',
   'DS_ESTADO_CIVIL', 'FE_NACIMIENTO', 'DS_PAIS_NACIMIENTO',
   'DS_NIVEL_ESTUDIOS', 'DS_PROFESION', 'DS_OCUPACION', 'DS_TIPO_VIVIENDA',
   'ID_CLIENTE', 'NO_PERSONAS_A_CARGO', 'FE_APERTURA', 'DK_PERSONA',
   'SK_RC', 'EDAD_ANIOS', 'ANTIGUEDAD_ANIOS', 'APERTURA_ANIOS',
   'Mes_Transact', 'fecha_dato_yearmonth', 'Edad_cat'],
  dtype='object')
```

```
In [ ]: """ Data frame para obtener la caracterizacion de los clientes """
```

```
df = pd.DataFrame()
```

```
In [ ]: df['transacciones'] = data.groupby('SK_CLIENTE')['CANTIDAD_TRANSACCION'].sum()
df['edad'] = data.groupby('SK_CLIENTE')['EDAD_ANIOS'].max()
```

```

df['antiguedad'] = data.groupby('SK_CLIENTE')['ANTIGUEDAD_ANIOS'].max()
df['apertura'] = data.groupby('SK_CLIENTE')['APERTURA_ANIOS'].min()
df['pais'] = data.groupby('SK_CLIENTE')['DS_PAIS_NACIMIENTO'].max()
df['ocupacion'] = data.groupby('SK_CLIENTE')['DS_OCCUPACION'].max()
df['tipo_vivienda'] = data.groupby('SK_CLIENTE')['DS_TIPO_VIVIENDA'].max()
df['estado_civil'] = data.groupby('SK_CLIENTE')['DS_ESTADO_CIVIL'].max()
df['nivel_estudios'] = data.groupby('SK_CLIENTE')['DS_NIVEL_ESTUDIOS'].max()
df['genero'] = data.groupby('SK_CLIENTE')['DS_GENERO'].max()
df['genero'] = df['genero'].map({'M': 1, 'F': 2, '-': 0})
df['no_personas_a_cargo'] = data.groupby('SK_CLIENTE')['NO_PERSONAS_A_CARGO'].max()
df['valor_mvto'] = data.groupby('SK_CLIENTE')['VALOR_MVTO'].mean().round(0)
df['cant_productos'] = data.groupby('SK_CLIENTE').NO_PRODUCTO.nunique()

```

In []: # validar la distribucion para las categorias nivel de estudio y ocupacion

```

for i in [df['nivel_estudios'], df['ocupacion']]:
    print(i.value_counts())

```

Primaria	7066
Profesional	5041
Tecnico Profesional	2368
Especializacion	2018
Sin estudio	915
-	28
Tecnologico	14
Maestria	9
Doctorado	3
Secundaria	2
Name: nivel_estudios, dtype: int64	
Pensionado	8508
Police Nacional	2977
Empleado Privado	2658
Docente	1746
Empleado Publico	504
Fuerzas Militares	369
Estudiante	304
Independiente	303
Hogar	95
Name: ocupacion, dtype: int64	

In []: #Se ajustan valores sobre los registros que no tienen una categoria definida
#(Nivel de estudios - Estado Civil - Tipo Vivienda - Ocupacion)

```

df.nivel_estudios = df.nivel_estudios.replace({'-': 'Sin estudio'})
df.estado_civil = df.estado_civil.replace({'-': 'Desconocido'})
df.tipo_vivienda = df.tipo_vivienda.replace({'-': 'Desconocido'})
df.ocupacion = df.ocupacion.replace({'-': 'Hogar'})
df.estado_civil = df.estado_civil.replace({'UNINONAibre': 'Union Libre'})

```

In []: #Variable que muestra si estado_civil se encuentran en pareja o no, disminuyendo la di
df["Con_Pareja"] = df["estado_civil"].replace({"Casado(a)": 1, "Soltero(a)": 0, "Union Libr

In []: #Se categoriza el nivel de estudios

```

df["Educacion"] = df["nivel_estudios"].replace({
    "Doctorado": "Postgrado", "Maestria": "Postgrado",
    "Primaria": "Academico", "Secundaria": "Academico",
    "Tecnico Profesional": "Tecnico", "Tecnologico": "Tecnico"
})

```

In []: #Reduce la categorizacion de la ocupacion del cliente

```

df.ocupacion = df.ocupacion.replace({
    "Police Nacional": "Fuerzas Militares", "Empleado Publico": "Fuerzas Militares",
    "Empleado Privado": "Fuerzas Militares", "Docente": "Fuerzas Militares",
    "Pensionado": "Fuerzas Militares", "Independiente": "Fuerzas Militares",
    "Estudiante": "Fuerzas Militares", "Fuerzas Militares": "Fuerzas Militares"
})

```

```
In [ ]: df
```

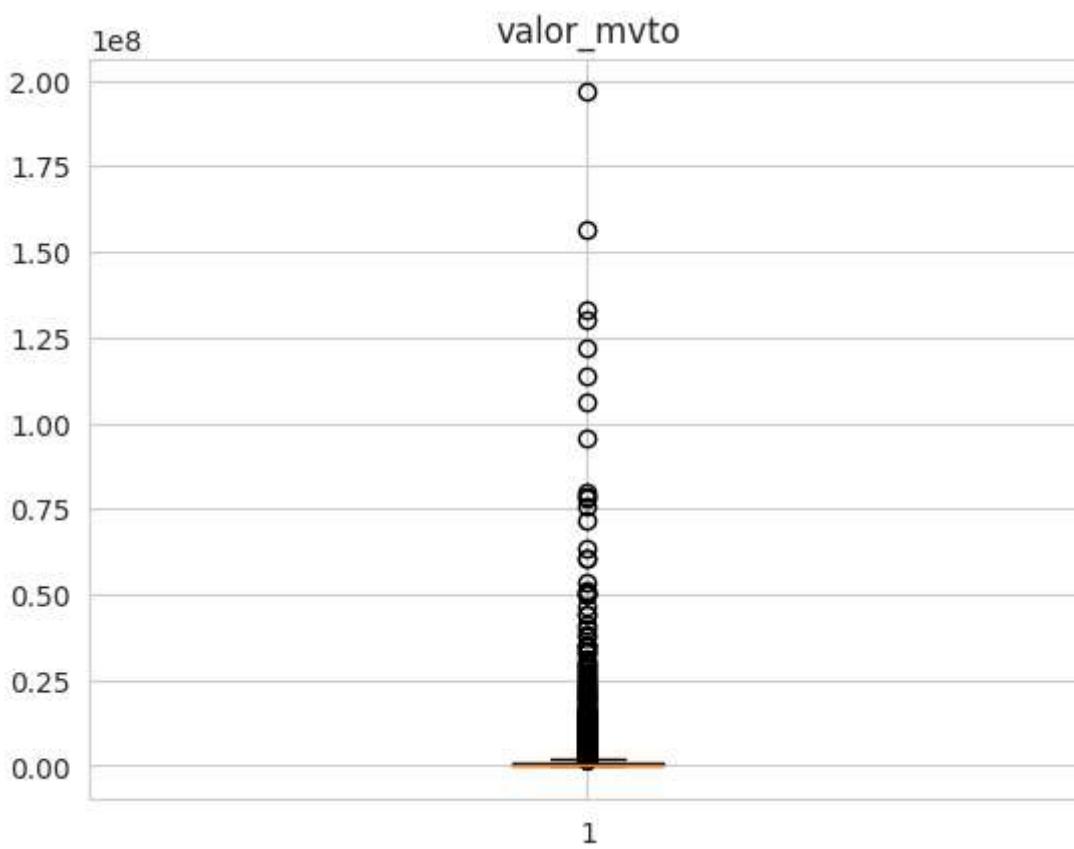
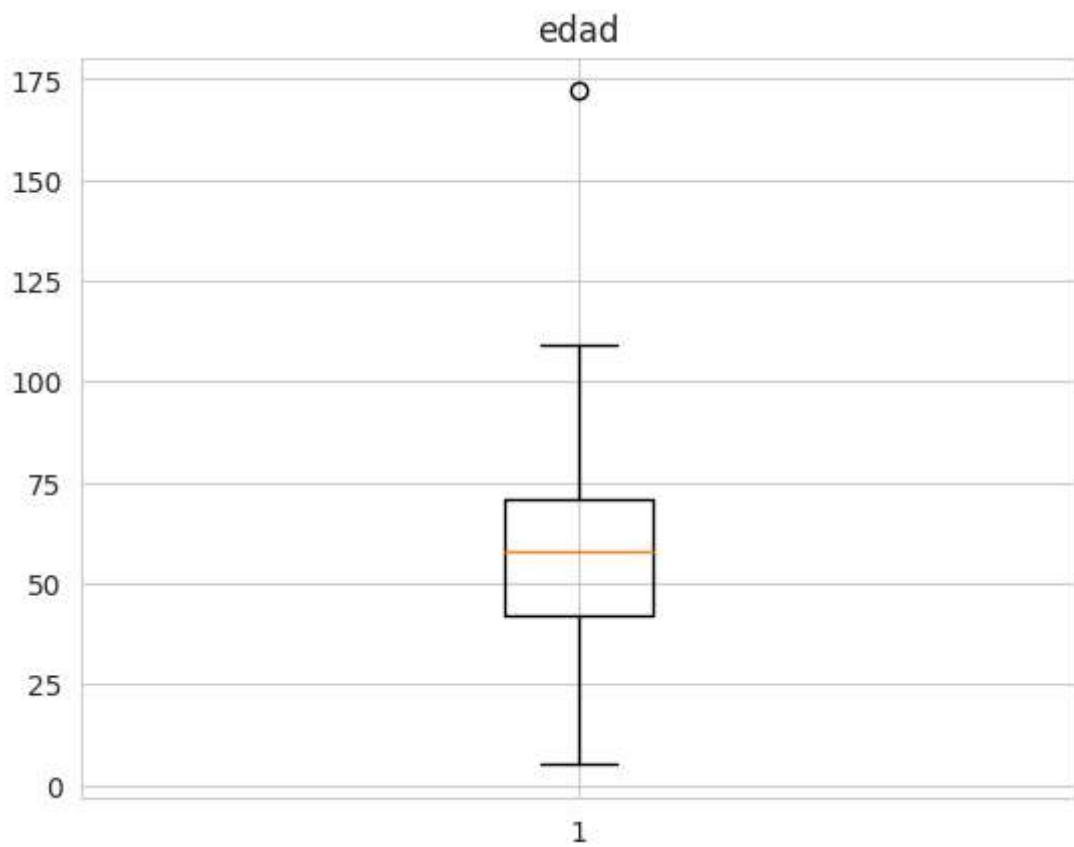
```
Out[ ]:      transacciones  edad  antiguedad  apertura      pais  ocupacion  tipo_vivienda  estado_civil
```

SK_CLIENTE

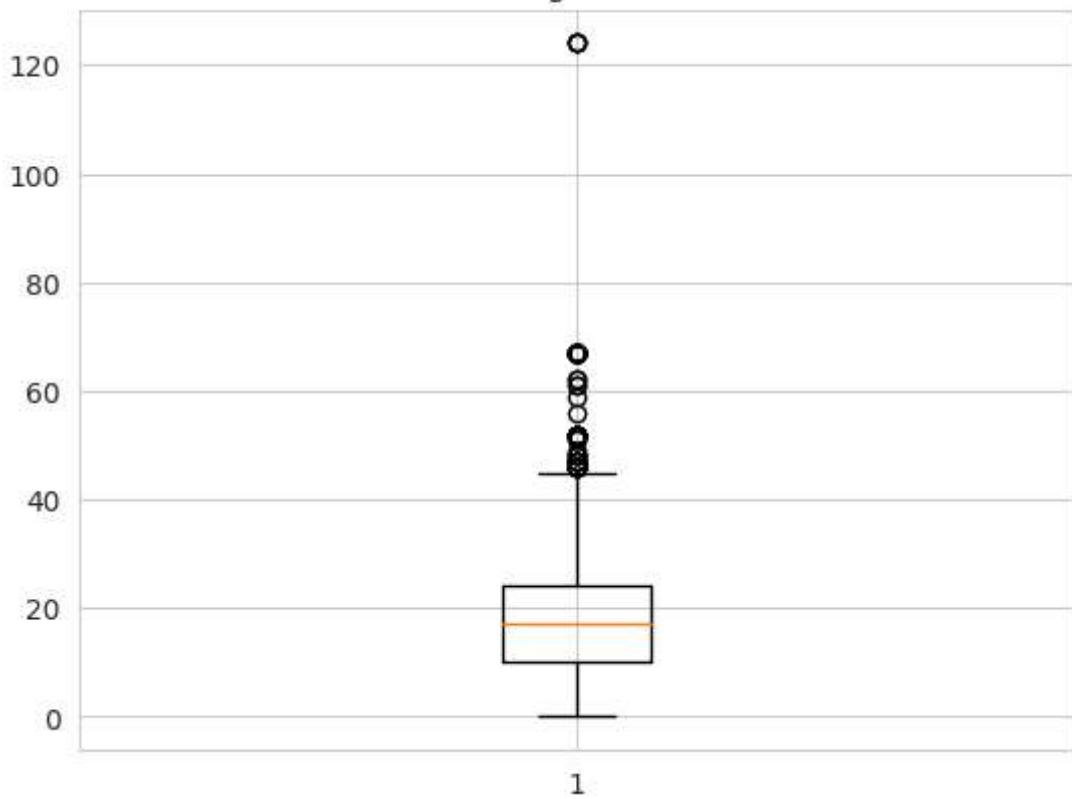
68937	1	80.0	26.0	26.0	Colombia	Pensionado	Familiar	Casado
83597	4	49.0	19.0	19.0	Colombia	Fuerzas Militares	Propia	Viudo
86570	6	63.0	25.0	5.0	Colombia	Pensionado	Propia	Soltero
191735	1	65.0	19.0	8.0	Colombia	Empleado	Familiar	Separado
192556	4	72.0	19.0	19.0	Colombia	Pensionado	Propia	Casado
...
4411791	2	42.0	5.0	5.0	Colombia	Fuerzas Militares	Familiar	Casado
4412445	1	49.0	22.0	22.0	Colombia	Pensionado	Propia	Union Libre
4415269	1	32.0	12.0	12.0	Colombia	Fuerzas Militares	Familiar	Soltero
4421550	1	32.0	7.0	7.0	Colombia	Empleado	Familiar	Soltero
4421738	1	60.0	24.0	22.0	Colombia	Empleado	Familiar	Soltero

17464 rows × 15 columns

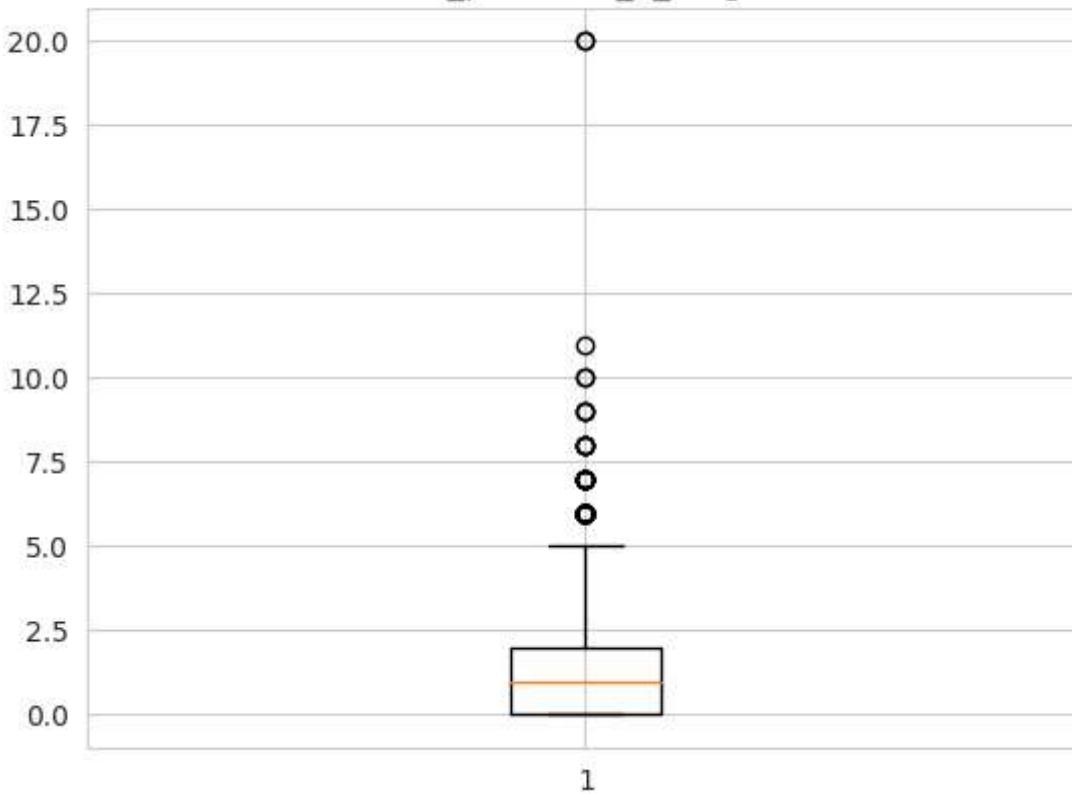
```
In [ ]: #Visualizamos la caracterizacion de los datos de 'edad', 'valor_saldo', 'valor_mvto',  
for i in ['edad', 'valor_mvto', 'antiguedad', 'no_personas_a_cargo']:  
    plt.boxplot(df[i])  
    plt.title(i)  
    plt.show()
```



antiguedad

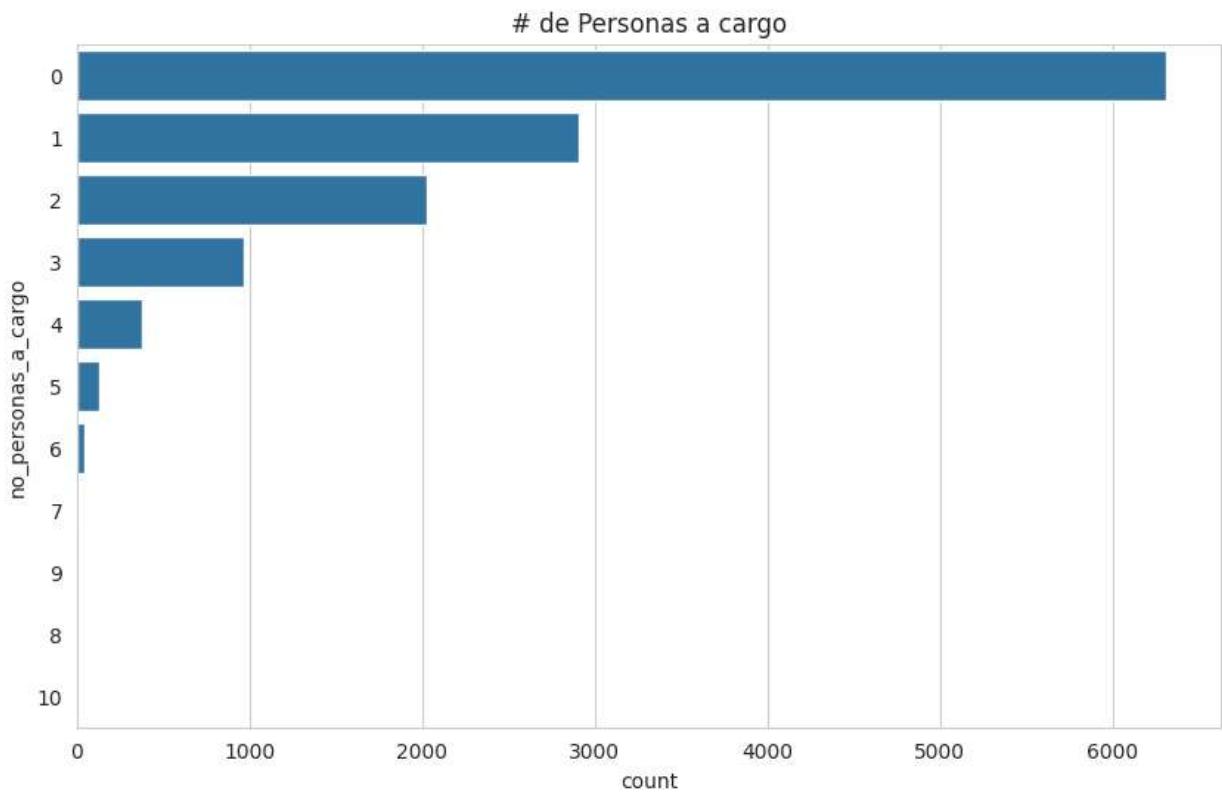


no_personas_a_cargo



```
In [ ]: df = df[df['edad'] <=120]
df = df[df['antiguedad'] <=46]
df = df[df['valor_mvto'] <= 857423]
df = df[df['no_personas_a_cargo'] <= 10]
```

```
In [ ]: plt.figure(figsize=(10,6))
sns.countplot(y="no_personas_a_cargo", data=df, order=df["no_personas_a_cargo"].value_
plt.title("# de Personas a cargo");
```



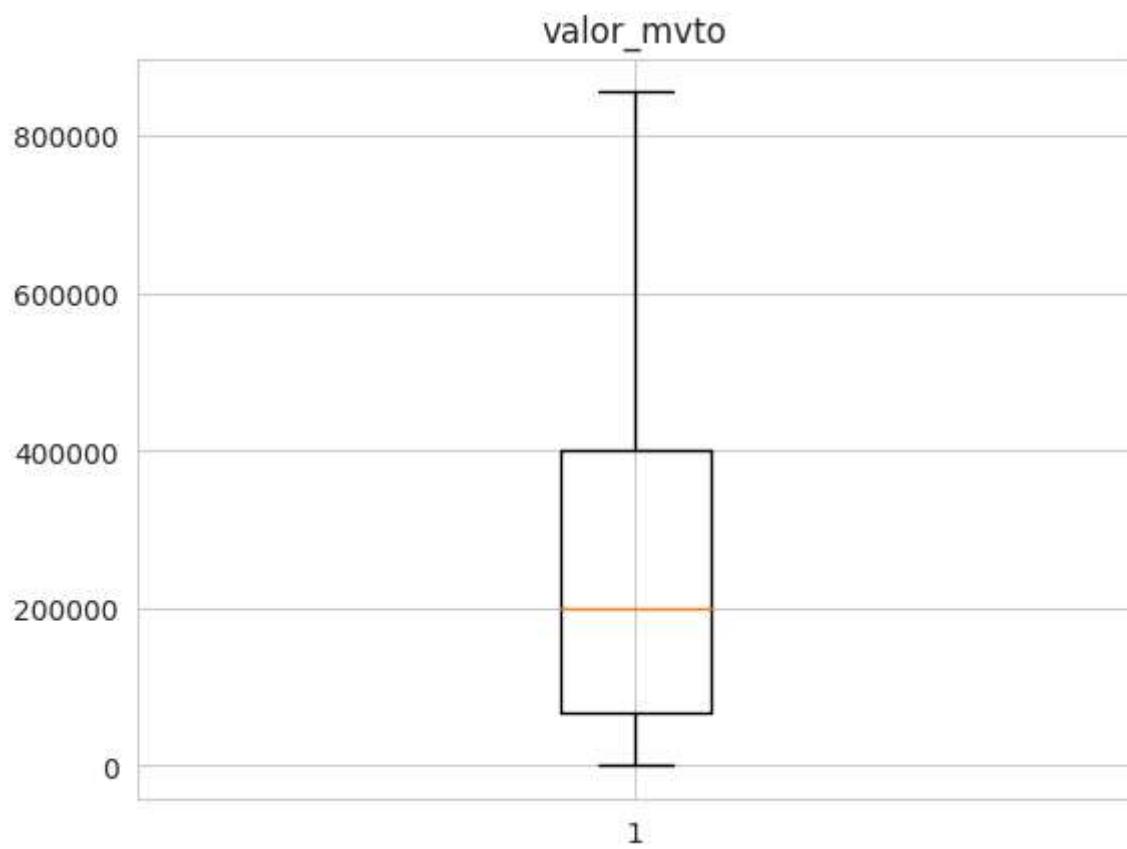
```
In [ ]: df
```

Out[]:

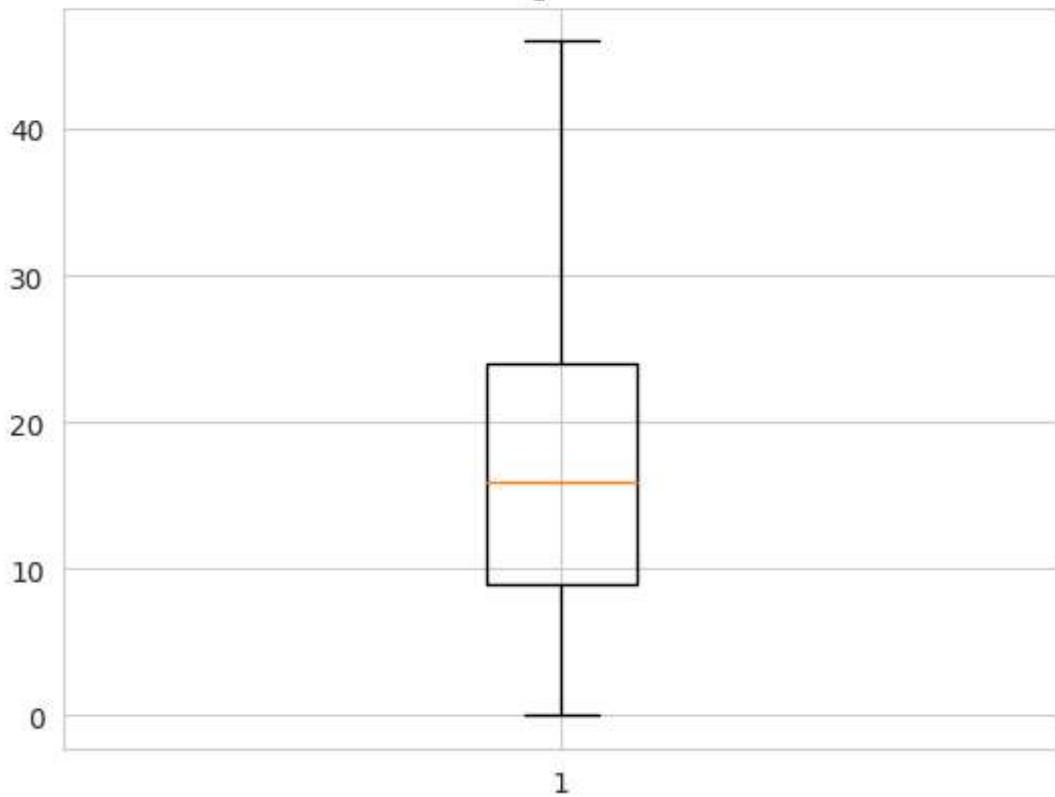
	transacciones	edad	antiguedad	apertura	pais	ocupacion	tipo_vivienda	estado_civil
SK_CLIENTE								
68937	1	80.0	26.0	26.0	Colombia	Pensionado	Familiar	Casado
83597	4	49.0	19.0	19.0	Colombia	Fuerzas Militares	Propia	Viudo
86570	6	63.0	25.0	5.0	Colombia	Pensionado	Propia	Soltero
197716	1	24.0	5.0	5.0	Colombia	Estudiante	Familiar	Soltero
204833	2	56.0	24.0	24.0	Colombia	Empleado	Propia	Casado
...
4410983	4	82.0	26.0	26.0	Colombia	Pensionado	Propia	Casado
4411791	2	42.0	5.0	5.0	Colombia	Fuerzas Militares	Familiar	Casado
4412445	1	49.0	22.0	22.0	Colombia	Pensionado	Propia	Union Li
4415269	1	32.0	12.0	12.0	Colombia	Fuerzas Militares	Familiar	Soltero
4421550	1	32.0	7.0	7.0	Colombia	Empleado	Familiar	Soltero

12772 rows × 15 columns

In []: #Visualizamos la caracterizacion de la 'edad', 'valor_saldo', 'valor_mvto', 'antiguedad'
for i in ['edad', 'valor_mvto', 'antiguedad']:
plt.boxplot(df[i])
plt.title(i)
plt.show()



antiguedad



Se va ha realizar validaciones para saber que tipo de variables se van a tomar en DBSCAN -
Análisis de correlación -Análisis de componentes principales (PCA) -Métodos de selección de
características -Validación cruzada

```
In [ ]: # Visualizar la matriz de correlación
correlation_matrix = df.corr()
print(correlation_matrix)
```

	transacciones	edad	antiguedad	apertura	genero	\
transacciones	1.000000	0.072801	-0.006681	-0.015776	0.041978	
edad		1.000000	0.324138	0.249936	0.200713	
antiguedad		-0.006681	0.324138	1.000000	0.837173	-0.079677
apertura		-0.015776	0.249936	0.837173	1.000000	-0.083640
genero		0.041978	0.200713	-0.079677	-0.083640	1.000000
no_personas_a_cargo		-0.001432	-0.042276	0.202088	0.219254	-0.183474
valor_mvto		0.165907	0.094722	0.062847	0.075661	0.034018
cant_productos		0.044626	0.026784	0.022750	-0.031237	0.011906
Con_Pareja		-0.007125	0.118889	0.217470	0.212701	-0.224750

	no_personas_a_cargo	valor_mvto	cant_productos	\
transacciones	-0.001432	0.165907	0.044626	
edad	-0.042276	0.094722	0.026784	
antiguedad	0.202088	0.062847	0.022750	
apertura	0.219254	0.075661	-0.031237	
genero	-0.183474	0.034018	0.011906	
no_personas_a_cargo	1.000000	0.013622	-0.008897	
valor_mvto	0.013622	1.000000	0.032489	
cant_productos	-0.008897	0.032489	1.000000	
Con_Pareja	0.285324	0.031355	-0.000678	

	Con_Pareja
transacciones	-0.007125
edad	0.118889
antiguedad	0.217470
apertura	0.212701
genero	-0.224750
no_personas_a_cargo	0.285324
valor_mvto	0.031355
cant_productos	-0.000678
Con_Pareja	1.000000

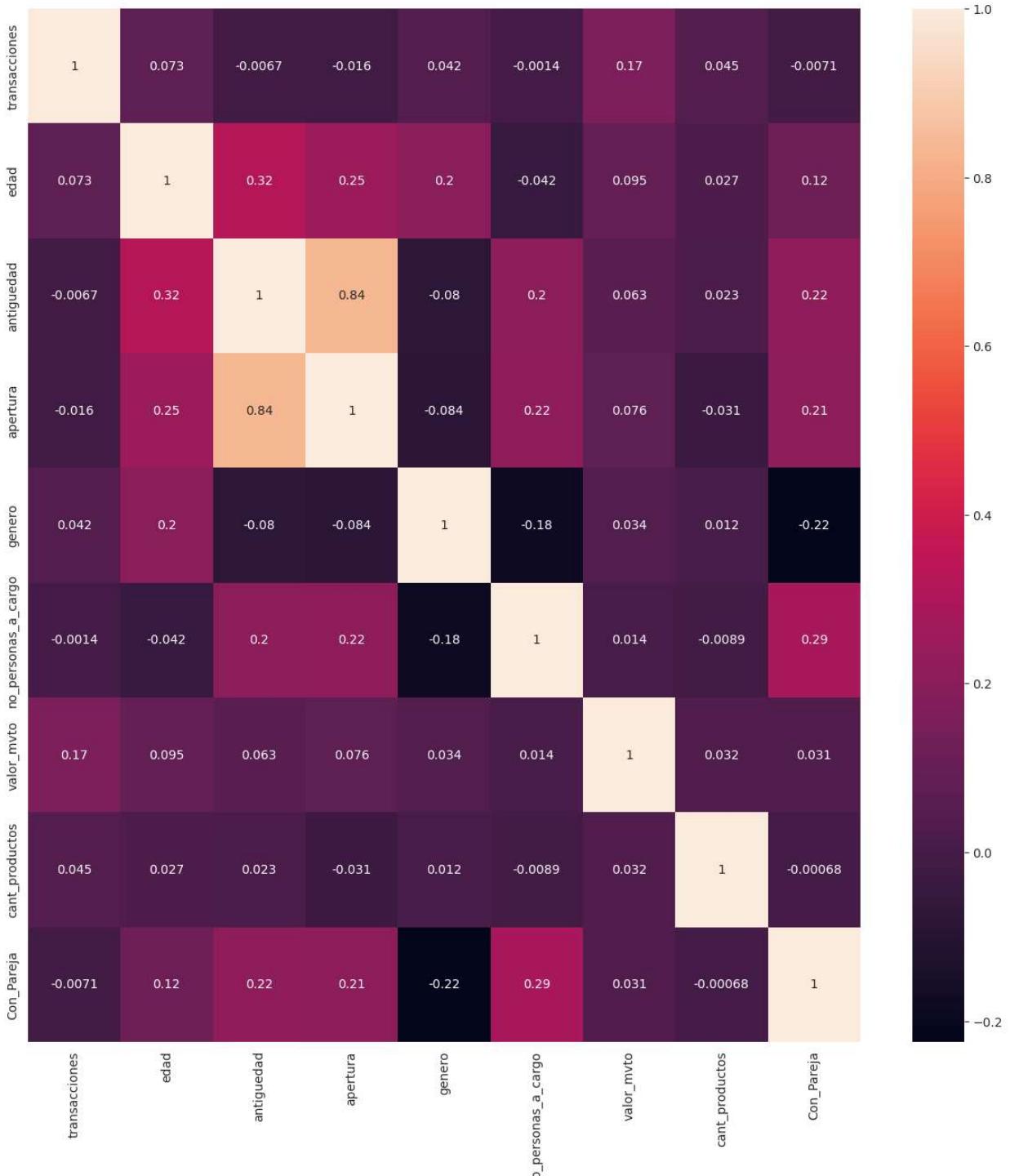
```
<ipython-input-44-78e773834dd6>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
    correlation_matrix = df.corr()
```

In []: #Correlacion de variables seleccionadas

```
"""Se valida que hay una relacion significante entre la antiguedad y la apertura se de
hay una correlacion moderada entre transacciones con: edad 0.073 , genero 0.42 y canti
edad correlacion moderada : antiguedad 0.32, genero 0.20, pareja 0.12
antiguedas moderada: pareja 0.22, no_personas a cargo 0.2
"""

correlations = df.corr()
f, ax = plt.subplots(figsize = (15,15))
sns.heatmap(correlations, annot=True)
correlations.round(2);
```

```
<ipython-input-45-9bc8e9f62eb8>:9: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
    correlations = df.corr()
```



```
In [ ]: df1=df
```

```
In [ ]: Y = df1[['transacciones','edad','genero','valor_mvto','cant_productos']]
```

```
In [ ]: # Definir los parámetros a ajustar
param_grid = {'eps': [0.1, 0.2, 0.3, 0.4, 0.5],
              'min_samples': [5, 10, 20, 50, 100]}
```

```
In [ ]: # Función de puntuación personalizada
def silhouette_scorer(estimator, Y):
    labels = estimator.fit_predict(Y)
    return silhouette_score(Y, labels)
```

```
In [ ]: dbSCAN = DBSCAN()
```

```
In [ ]: grid_search = GridSearchCV(dbSCAN, param_grid, scoring=silhouette_scorer)
grid_search.fit(Y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non-finite: [nan nan nan nan nan nan n  
an nan nan]  
    warnings.warn(  
Out[ ]: ▶ GridSearchCV  
      ▶ estimator: DBSCAN  
        ▶ DBSCAN
```

```
In [ ]: # Imprimir Los mejores parámetros y el coeficiente de silueta  
print("Mejores parámetros:", grid_search.best_params_)  
print("Coeficiente de silueta:", silhouette_score(Y, grid_search.best_estimator_.label)  
Mejores parámetros: {'eps': 0.1, 'min_samples': 5}  
Coeficiente de silueta: -0.7420871600034935
```

Un coeficiente de silueta de -0.74208 indica que el agrupamiento producido por el algoritmo DBSCAN no es satisfactorio. Este valor negativo sugiere que hay una gran superposición entre los clústeres o que las muestras han sido asignadas incorrectamente a los clústeres.

Algunas conclusiones que se pueden extraer de un coeficiente de silueta negativo tan bajo son:

Superposición significativa entre clústeres: Los clústeres obtenidos por el algoritmo DBSCAN se superponen entre sí, lo que indica que el algoritmo no ha podido separar claramente las diferentes clases o grupos en los datos.

Asignaciones incorrectas: Es posible que algunas muestras estén asignadas incorrectamente a clústeres que no reflejan su verdadera pertenencia. Esto puede deberse a una selección inadecuada de los parámetros del algoritmo DBSCAN (epsilon y min_samples) o a la naturaleza intrínseca de los datos.

Distribución no densa o heterogénea de los datos: La distribución de los datos puede ser tan dispersa o heterogénea que el algoritmo DBSCAN no es capaz de encontrar estructuras significativas de clústeres.

Falta de separación entre clústeres: Los clústeres pueden estar tan cerca entre sí que DBSCAN no puede distinguir claramente entre ellos, lo que resulta en una mala calidad del agrupamiento.

un coeficiente de silueta tan bajo sugiere que el algoritmo DBSCAN no ha logrado encontrar una estructura de clústeres clara y coherente en los datos. Es posible que necesites ajustar los parámetros del algoritmo, explorar diferentes técnicas de agrupamiento o considerar la posibilidad de que los datos no tengan una estructura de clústeres clara y definida.

```
In [ ]: X = df1[['transacciones', 'edad', 'genero', 'valor_mvto', 'cant_productos']]
```

```
In [ ]: scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X)
```

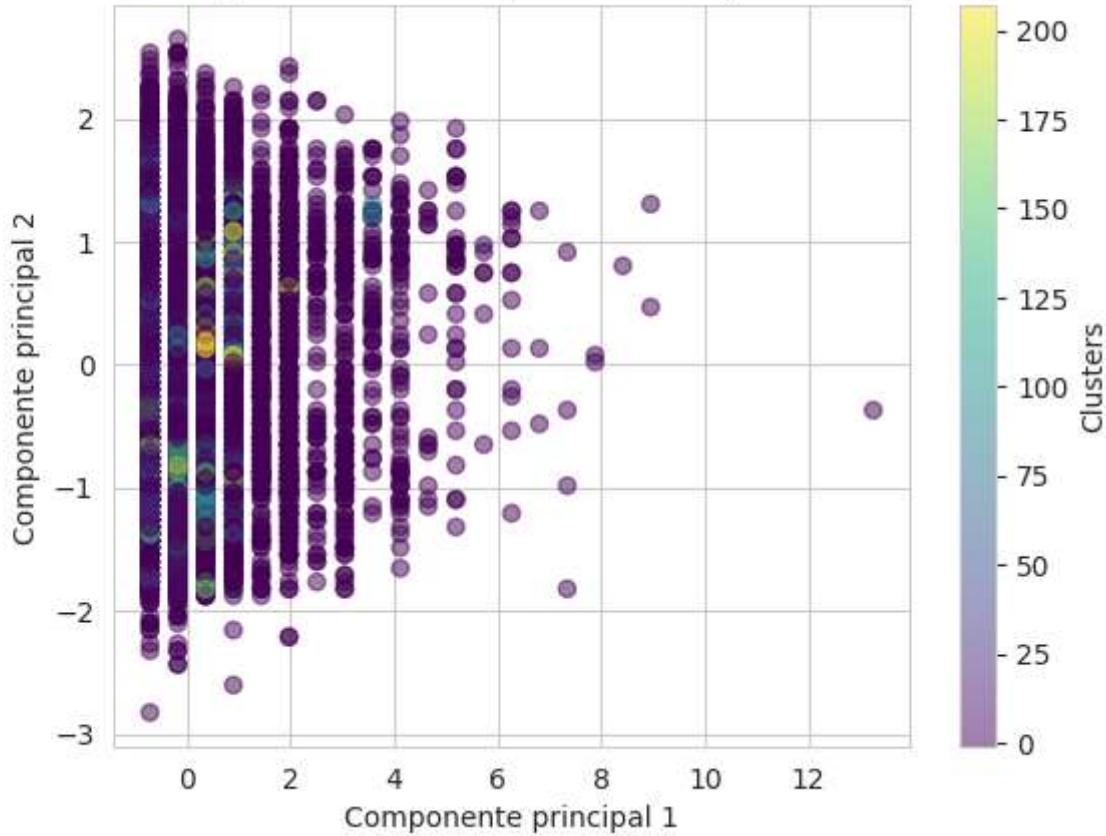
```
In [ ]: # Aplicar DBSCAN
dbscan = DBSCAN(eps=0.1, min_samples=5) # Puedes ajustar eps y min_samples según tus
clusters = dbscan.fit_predict(X_scaled)
print(clusters)

[ 0 -1 -1 ...  1  1 21]
```

```
In [ ]: # Visualizar Los resultados
#plt.scatter(X_scaled[:,0], X_scaled[:,1], c=clusters, cmap='viridis')
#plt.xlabel('Feature 1')
#plt.ylabel('Feature 2')
#plt.title('DBSCAN Clustering')
#plt.colorbar(label='Cluster')
#plt.show()

# Visualizar Los resultados de DBSCAN
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=clusters, cmap='viridis', alpha=0.5)
plt.xlabel('Componente principal 1')
plt.ylabel('Componente principal 2')
plt.title('DBSCAN en el espacio de los dos primeros componentes principales')
plt.colorbar(label='Clusters')
plt.grid(True)
plt.show()
```

DBSCAN en el espacio de los dos primeros componentes principales



```
In [ ]: # Calcula el coeficiente de silueta
silhouette_avg = silhouette_score(X_scaled, clusters)
print("Coeficiente de Silueta:", silhouette_avg)
```

Coeficiente de Silueta: -0.40310872249607654

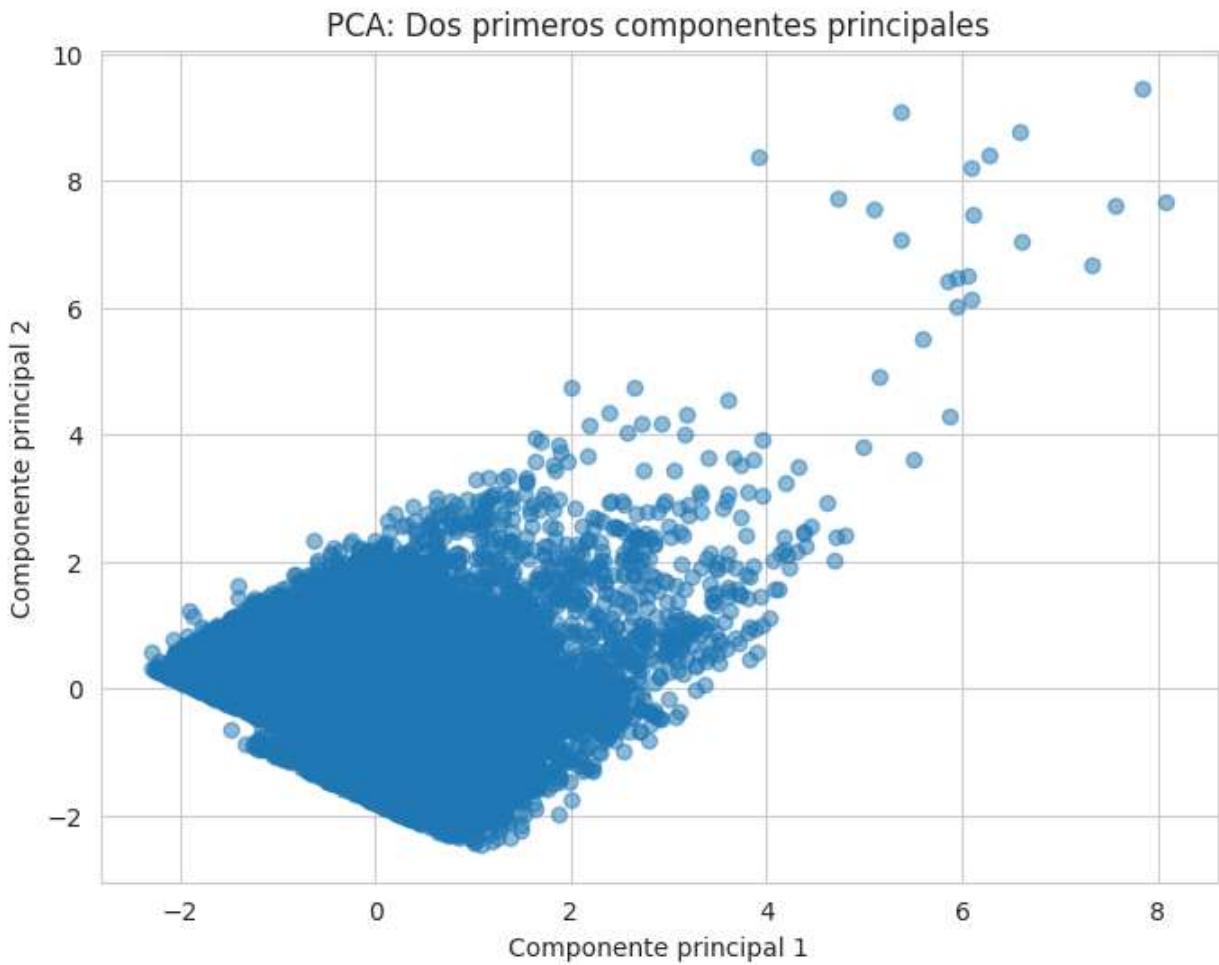
```
In [ ]: dfo= df  
In [ ]: columns_for_pca = ['transacciones','edad','genero','valor_mvto','cant_productos']  
In [ ]: x2 = dfo[columns_for_pca]  
In [ ]: x2
```

```
Out[ ]:      transacciones  edad  genero  valor_mvto  cant_productos
```

SK_CLIENTE					
68937	1	80.0	1	600000.0	1
83597	4	49.0	1	119928.0	1
86570	6	63.0	2	118807.0	1
197716	1	24.0	1	3000.0	1
204833	2	56.0	2	76061.0	1
...
4410983	4	82.0	2	349481.0	1
4411791	2	42.0	2	189758.0	1
4412445	1	49.0	1	20000.0	1
4415269	1	32.0	1	35500.0	1
4421550	1	32.0	2	205000.0	1

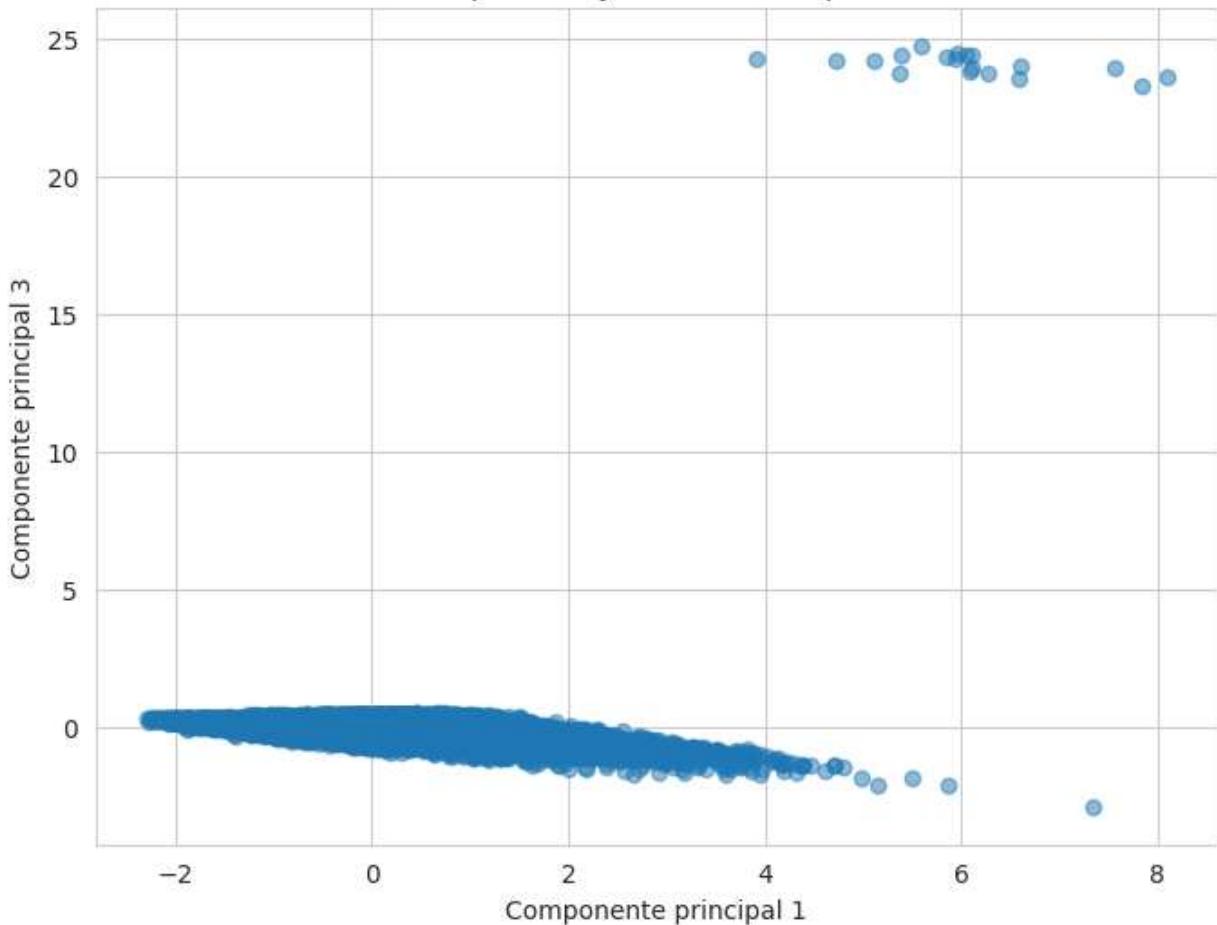
12772 rows × 5 columns

```
In [ ]: scaler = StandardScaler()  
X_scaled = scaler.fit_transform(x2)  
  
In [ ]: # Aplicar PCA  
pca = PCA(n_components=4) # Puedes ajustar el número de componentes principales según  
X_pca = pca.fit_transform(X_scaled)  
  
In [ ]: pca_df = pd.DataFrame(data=X_pca, columns=['Componente principal 1', 'Componente prin  
  
In [ ]: print("Varianza explicada por cada componente:")  
print(pca.explained_variance_ratio_)  
  
Varianza explicada por cada componente:  
[0.26368942 0.21427666 0.1967398 0.16771738]  
  
In [ ]: # Graficar Los datos en el espacio de Los dos primeros componentes principales  
plt.figure(figsize=(8, 6))  
plt.scatter(pca_df['Componente principal 1'], pca_df['Componente principal 2'], alpha=0.5)  
plt.xlabel('Componente principal 1')  
plt.ylabel('Componente principal 2')  
plt.title('PCA: Dos primeros componentes principales')  
plt.grid(True)  
plt.show()
```



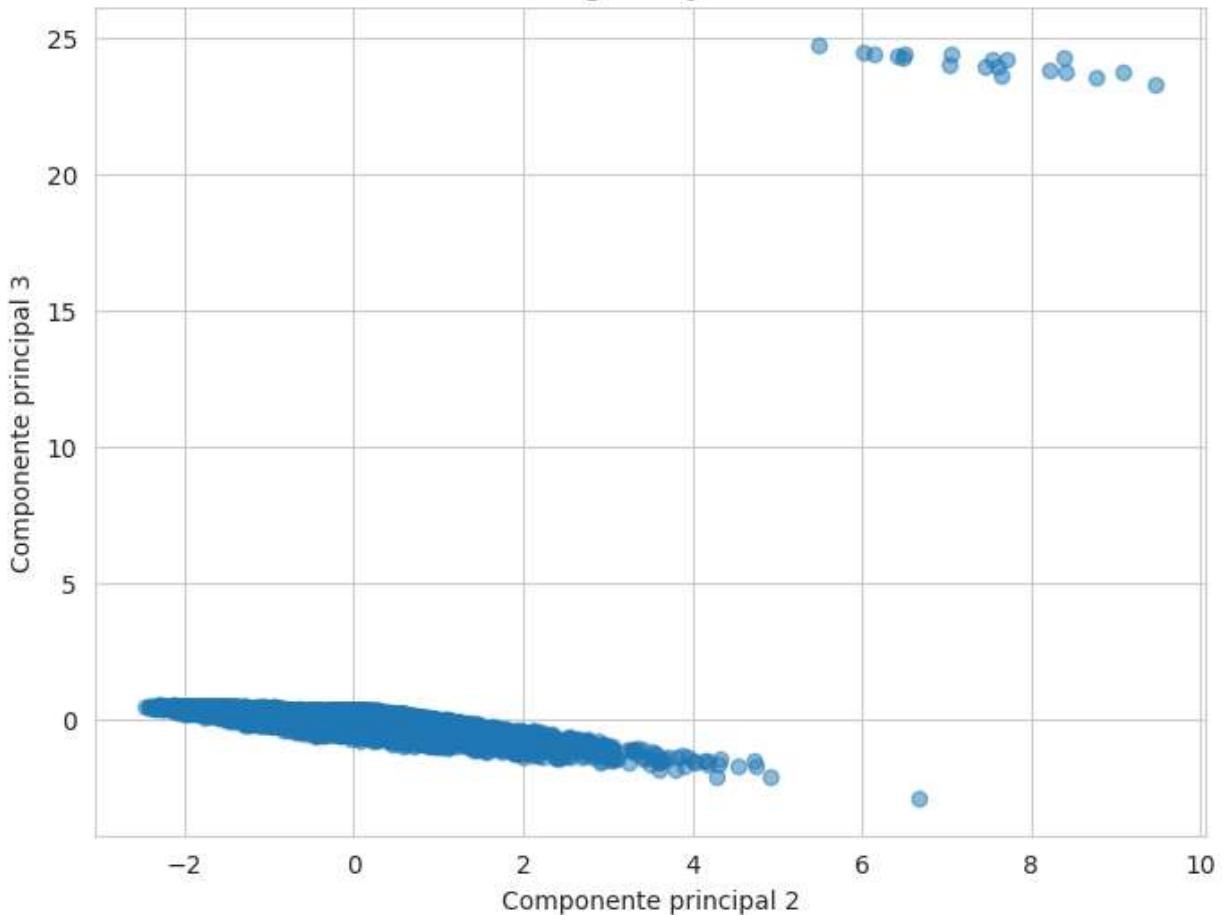
```
In [ ]: # Graficar los datos en el espacio de los dos primeros componentes principales
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['Componente principal 1'], pca_df['Componente principal 3'], alpha=0.5)
plt.xlabel('Componente principal 1')
plt.ylabel('Componente principal 3')
plt.title('PCA: el primero y el tercer componente ')
plt.grid(True)
plt.show()
```

PCA: el primero y el tercer componente



```
In [ ]: # Graficar los datos en el espacio de los dos primeros componentes principales
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['Componente principal 2'], pca_df['Componente principal 3'], alpha=0.5)
plt.xlabel('Componente principal 2')
plt.ylabel('Componente principal 3')
plt.title('PCA:El segundo y el tercero')
plt.grid(True)
plt.show()
```

PCA: El segundo y el tercero



```
In [ ]: k = 4 # Por ejemplo, seleccionamos los cuatro primeros componentes principales  
selected_components = pca.components_[:k]
```

```
In [ ]: # Proyectar los datos originales en el espacio de los k componentes principales  
X_projected = X_scaled.dot(selected_components.T)
```

```
In [ ]: # Definir los parámetros a ajustar  
param_grid = {'eps': [0.1, 0.2, 0.3, 0.4, 0.5],  
              'min_samples': [5, 10, 20, 50, 100]}
```

```
In [ ]: # Función de puntuación personalizada  
def silhouette_scorer(estimator, X_projected):  
    labels = estimator.fit_predict(X_projected)  
    return silhouette_score(X_projected, labels)
```

```
In [ ]: dbscan = DBSCAN()
```

```
In [ ]: grid_search = GridSearchCV(dbscan, param_grid, scoring=silhouette_scorer)  
grid_search.fit(X_projected)
```

```
File "/usr/local/lib/python3.10/dist-packages/sklearn/metrics/cluster/_unsupervise
d.py", line 33, in check_number_of_labels
    raise ValueError(
ValueError: Number of labels is 1. Valid values are 2 to n_samples - 1 (inclusive)

    warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserW
arning: One or more of the test scores are non-finite: [-0.20818526 -0.39057399
nan         nan         nan -0.26607481
-0.20332839 -0.19926618         nan         nan -0.18737293 -0.23419213
-0.12290206 -0.156748         nan -0.1022601 -0.14339569 -0.10222136
-0.06948892 -0.06123969 -0.02759922  0.03497174 -0.05738916  0.1559326
 0.00836653]
    warnings.warn(
```

Out[]:

- ▶ **GridSearchCV**
- ▶ **estimator: DBSCAN**
 - ▶ **DBSCAN**

In []:

```
# Imprimir los mejores parámetros y el coeficiente de silueta
print("Mejores parámetros:", grid_search.best_params_)
print("Coeficiente de silueta:", silhouette_score(Y, grid_search.best_estimator_.label]
```

Mejores parámetros: {'eps': 0.5, 'min_samples': 50}
Coeficiente de silueta: 0.21336869975915854

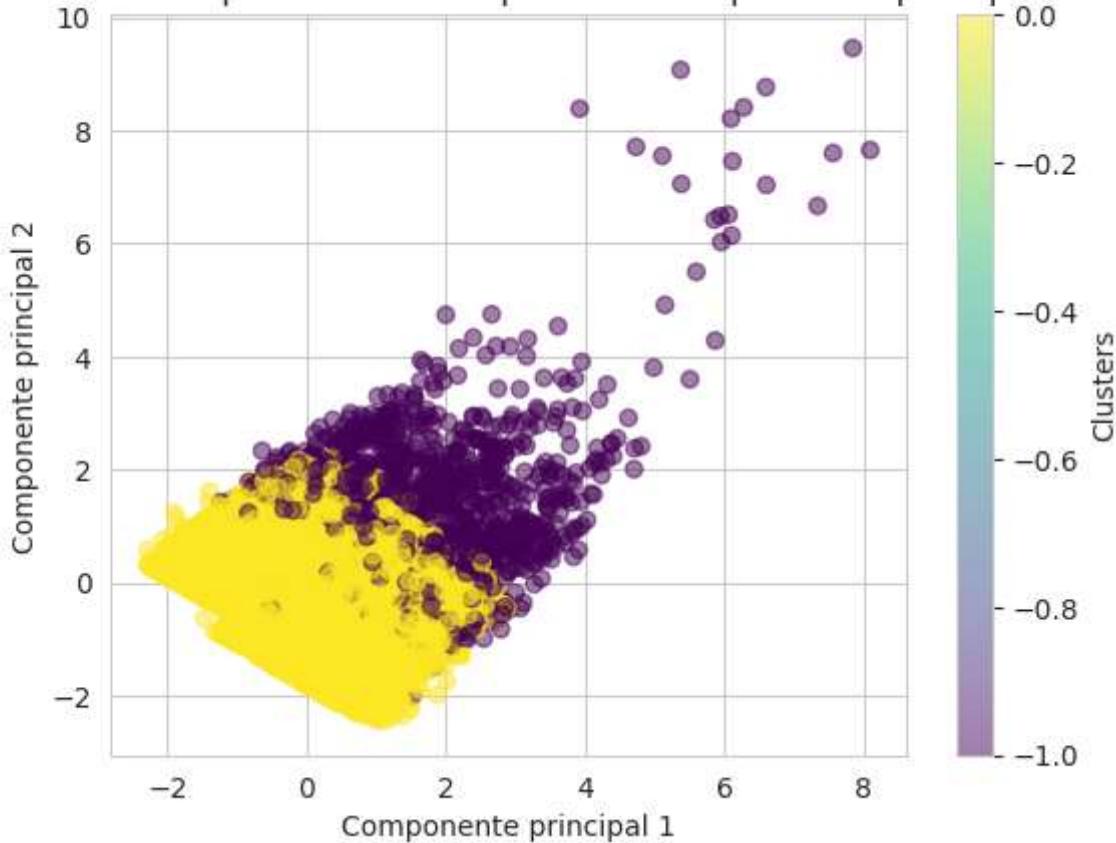
In []:

```
# Aplicar DBSCAN en el espacio de los componentes principales
dbscan = DBSCAN(eps=0.5, min_samples=50)
clusters = dbscan.fit_predict(X_projected)
```

In []:

```
# Visualizar los resultados de DBSCAN
plt.scatter(X_projected[:, 0], X_projected[:, 1], c=clusters, cmap='viridis', alpha=0.6)
plt.xlabel('Componente principal 1')
plt.ylabel('Componente principal 2')
plt.title('DBSCAN en el espacio de los dos primeros componentes principales')
plt.colorbar(label='Clusters')
plt.grid(True)
plt.show()
```

DBSCAN en el espacio de los dos primeros componentes principales



```
In [ ]: # Calcula el coeficiente de silueta  
silhouette_avg = silhouette_score(X_projected, clusters)  
print("Coeficiente de Silueta:", silhouette_avg)
```

Coeficiente de Silueta: 0.43211090185663636

Un coeficiente de silueta de 0.432 es un valor positivo y relativamente alto, lo que sugiere que el clustering producido por el algoritmo DBSCAN es bastante satisfactorio y muestra una buena separación entre los clústeres. Aquí hay algunas interpretaciones basadas en este valor:

Buena separación entre clústeres: El valor positivo indica que las muestras dentro de cada clúster están bien agrupadas y separadas de las muestras en otros clústeres. Esto sugiere que DBSCAN ha identificado claramente estructuras de clústeres distintas en los datos.

Clústeres bien definidos: Los clústeres son probablemente bien definidos y tienen una buena cohesión interna, lo que significa que las muestras dentro de cada clúster son muy similares entre sí en comparación con las muestras en otros clústeres.

Poca superposición entre clústeres: El coeficiente de silueta relativamente alto sugiere que hay poca superposición entre los clústeres, lo que indica una buena separación y definición de los mismos.

Alto grado de homogeneidad y cohesión: Las muestras dentro de cada clúster son más similares entre sí que con las muestras en otros clústeres, lo que sugiere una alta homogeneidad y cohesión dentro de los clústeres.

Un coeficiente de silueta de 0.432 indica que el algoritmo DBSCAN ha logrado identificar clústeres de manera efectiva y que estos clústeres muestran una buena separación y cohesión entre sí. Esto sugiere que el clustering es satisfactorio y proporciona una representación significativa de la estructura subyacente de los datos.

```
In [ ]: df2 = df
```

```
In [ ]: # Índice de Davies-Bouldin
db_score = davies_bouldin_score(X_projected, clusters)
print("Índice de Davies-Bouldin:", db_score)
```

Índice de Davies-Bouldin: 1.4446590340864747

El Índice de Davies-Bouldin (DB) es una métrica de validación interna que se utiliza para evaluar la calidad de un agrupamiento. Cuanto menor sea el valor del índice de Davies-Bouldin, mejor será el agrupamiento.

El resultado obtenido, 1.4446590340864747, indica que el agrupamiento tiene un valor medio de Davies-Bouldin. La interpretación de este valor depende del contexto y de la comparación con otros resultados. Aquí hay algunas interpretaciones generales:

Si el índice de Davies-Bouldin es cercano a 0, significa que los clústeres están bien separados y tienen una buena cohesión interna. Este es el escenario ideal.

Si el índice de Davies-Bouldin es mayor que 1, indica que los clústeres no están bien separados y pueden estar superpuestos. Un valor alto sugiere que el agrupamiento no es tan bueno y que los clústeres pueden no ser representativos o relevantes.

Un valor de 1.44 indica que el agrupamiento tiene una calidad media en términos de separación entre clústeres y cohesión interna.

```
In [ ]: # Criterio de Calinski-Harabasz
ch_score = calinski_harabasz_score(X_projected, clusters)
print("Criterio de Calinski-Harabasz:", ch_score)
```

Criterio de Calinski-Harabasz: 2258.8312242131137

El Criterio de Calinski-Harabasz (CH) es una métrica de validación interna que se utiliza para evaluar la calidad de un agrupamiento. Cuanto mayor sea el valor del criterio de Calinski-Harabasz, mejor será el agrupamiento.

El resultado obtenido, 2258.8312242131137, indica una puntuación bastante alta en el criterio de Calinski-Harabasz. La interpretación de este valor es la siguiente:

Un valor alto del criterio de Calinski-Harabasz sugiere que los clústeres están bien separados entre sí y tienen una buena cohesión interna. Esto significa que los puntos dentro de cada clúster son similares entre sí y que los clústeres están claramente definidos y separados. Un valor de 2258.83 en el criterio de Calinski-Harabasz indica que el agrupamiento tiene una buena calidad en términos de separación entre clústeres y cohesión interna.

```
In [ ]: # Añadir las etiquetas de clúster al DataFrame original  
dfo['cluster'] = clusters
```

```
In [ ]: dfo
```

```
Out[ ]:      transacciones  edad  antiguedad  apertura      pais  ocupacion  tipo_vivienda  estado_c  
SK_CLIENTE  
 68937          1    80.0       26.0     26.0  Colombia  Pensionado    Familiar   Casado  
 83597          4    49.0       19.0     19.0  Colombia  Fuerzas Militares  Propia  Viudo  
 86570          6    63.0       25.0      5.0  Colombia  Pensionado  Propia  Soltero  
 197716         1    24.0       5.0      5.0  Colombia  Estudiante  Familiar  Soltero  
 204833         2    56.0       24.0     24.0  Colombia  Empleado   Propia  Casado  
 ...           ...     ...       ...     ...       ...       ...       ...       ...  
 4410983        4    82.0       26.0     26.0  Colombia  Pensionado  Propia  Casado  
 4411791        2    42.0       5.0      5.0  Colombia  Fuerzas Militares  Familiar  Casado  
 4412445        1    49.0       22.0     22.0  Colombia  Pensionado  Propia  Union Li  
 4415269        1    32.0       12.0     12.0  Colombia  Fuerzas Militares  Familiar  Soltero  
 4421550        1    32.0       7.0      7.0  Colombia  Empleado   Familiar  Soltero
```

12772 rows × 16 columns

```
In [ ]: # Crear un diccionario para almacenar Los DataFrames de cada clúster  
cluster_datasets = {}
```

```
# Iterar sobre cada clúster y extraer Los datos correspondientes  
for cluster_label in np.unique(clusters):
```

```
    cluster_data = df[df['cluster'] == cluster_label]  
    cluster_datasets[cluster_label] = cluster_data
```

```
In [ ]: cluster_datasets
```

```

Out[ ]: {-1: transacciones edad antiguedad apertura pais ocupacion \
SK_CLIENTE
86570 6 63.0 25.0 5.0 Colombia Pensionado
214784 5 62.0 4.0 4.0 Colombia Pensionado
215177 5 44.0 17.0 17.0 Colombia Estudiante
216798 5 47.0 24.0 20.0 Colombia Empleado
221893 8 45.0 15.0 15.0 Colombia Empleado
...
... ...
... ...
4396773 4 84.0 37.0 23.0 Colombia Pensionado
4397027 5 41.0 4.0 4.0 Colombia Empleado
4397739 5 44.0 7.0 7.0 Colombia Empleado
4397880 3 49.0 20.0 4.0 Colombia Empleado
4398949 6 52.0 18.0 9.0 Colombia Empleado

tipo_vivienda estado_civil nivel_estudios genero \
SK_CLIENTE
86570 Propia Soltero(a) Profesional 2
214784 Propia Union Libre Profesional 1
215177 Propia Union Libre Especializacion 2
216798 Familiar Casado(a) Profesional 2
221893 Arrendada Casado(a) Especializacion 2
...
... ...
... ...
4396773 Propia Casado(a) Primaria 1
4397027 Propia Union Libre Especializacion 1
4397739 Arrendada Separado(a) Especializacion 1
4397880 Familiar Casado(a) Especializacion 2
4398949 Propia Casado(a) Profesional 2

no_personas_a_cargo valor_mvto cant_productos Con_Pareja \
SK_CLIENTE
86570 0 118807.0 1 0
214784 3 85742.0 1 1
215177 0 231058.0 1 1
216798 3 571035.0 1 1
221893 1 463925.0 1 1
...
... ...
... ...
4396773 1 188236.0 2 1
4397027 1 183014.0 1 1
4397739 0 202367.0 1 0
4397880 3 717000.0 2 1
4398949 1 469436.0 1 1

Educacion cluster
SK_CLIENTE
86570 Profesional -1
214784 Profesional -1
215177 Postgrado -1
216798 Profesional -1
221893 Postgrado -1
...
... ...
4396773 Academic -1
4397027 Postgrado -1
4397739 Postgrado -1
4397880 Postgrado -1
4398949 Profesional -1

[1313 rows x 16 columns],
0: transacciones edad antiguedad apertura pais \
SK_CLIENTE
68937 1 80.0 26.0 26.0 Colombia

```

83597	4	49.0	19.0	19.0	Colombia
197716	1	24.0	5.0	5.0	Colombia
204833	2	56.0	24.0	24.0	Colombia
205535	1	36.0	13.0	13.0	Colombia
...
4410983	4	82.0	26.0	26.0	Colombia
4411791	2	42.0	5.0	5.0	Colombia
4412445	1	49.0	22.0	22.0	Colombia
4415269	1	32.0	12.0	12.0	Colombia
4421550	1	32.0	7.0	7.0	Colombia

SK_CLIENTE	ocupacion	tipo_vivienda	estado_civil	nivel_estudios	\
68937	Pensionado	Familiar	Casado(a)	Profesional	
83597	Fuerzas Militares	Propia	Viudo(a)	Especializacion	
197716	Estudiante	Familiar	Soltero(a)	Primaria	
204833	Empleado	Propia	Casado(a)	Primaria	
205535	Docente	Arrendada	Casado(a)	Profesional	
...
4410983	Pensionado	Propia	Casado(a)	Primaria	
4411791	Fuerzas Militares	Familiar	Casado(a)	Primaria	
4412445	Pensionado	Propia	Union Libre	Primaria	
4415269	Fuerzas Militares	Familiar	Soltero(a)	Primaria	
4421550	Empleado	Familiar	Soltero(a)	Profesional	

SK_CLIENTE	genero	no_personas_a_cargo	valor_mvto	cant_productos	\
68937	1	2	600000.0	1	
83597	1	5	119928.0	1	
197716	1	0	3000.0	1	
204833	2	0	76061.0	1	
205535	1	2	239200.0	1	
...
4410983	2	0	349481.0	1	
4411791	2	0	189758.0	1	
4412445	1	0	20000.0	1	
4415269	1	0	35500.0	1	
4421550	2	0	205000.0	1	

SK_CLIENTE	Con_Pareja	Educacion	cluster	
68937	1	Profesional	0	
83597	0	Postgrado	0	
197716	0	Academico	0	
204833	1	Academico	0	
205535	1	Profesional	0	
...	
4410983	1	Academico	0	
4411791	1	Academico	0	
4412445	1	Academico	0	
4415269	0	Academico	0	
4421550	0	Profesional	0	

[11459 rows x 16 columns]}

```
In [ ]: cluster_label = 0
cluster_data = cluster_datasets[cluster_label]
```

```

print(cluster_data.head())

```

		transacciones	edad	antiguedad	apertura	pais	\
SK_CLIENTE							
68937		1	80.0	26.0	26.0	Colombia	
83597		4	49.0	19.0	19.0	Colombia	
197716		1	24.0	5.0	5.0	Colombia	
204833		2	56.0	24.0	24.0	Colombia	
205535		1	36.0	13.0	13.0	Colombia	

		ocupacion	tipo_vivienda	estado_civil	nivel_estudios	\
SK_CLIENTE						
68937		Pensionado	Familiar	Casado(a)	Profesional	
83597		Fuerzas Militares	Propia	Viudo(a)	Especializacion	
197716		Estudiante	Familiar	Soltero(a)	Primaria	
204833		Empleado	Propia	Casado(a)	Primaria	
205535		Docente	Arrendada	Casado(a)	Profesional	

		genero	no_personas_a_cargo	valor_mvto	cant_productos	\
SK_CLIENTE						
68937		1		2	600000.0	
83597		1		5	119928.0	
197716		1		0	3000.0	
204833		2		0	76061.0	
205535		1		2	239200.0	

		Con_Pareja	Educacion	cluster
SK_CLIENTE				
68937		1	Profesional	0
83597		0	Postgrado	0
197716		0	Academico	0
204833		1	Academico	0
205535		1	Profesional	0

```

In [ ]: #
cluster_label = -1
cluster_data2 = cluster_datasets[cluster_label]

print(cluster_data2.head())

```

	transacciones	edad	antiguedad	apertura	pais	ocupacion	\
SK_CLIENTE							
86570	6	63.0	25.0	5.0	Colombia	Pensionado	
214784	5	62.0	4.0	4.0	Colombia	Pensionado	
215177	5	44.0	17.0	17.0	Colombia	Estudiante	
216798	5	47.0	24.0	20.0	Colombia	Empleado	
221893	8	45.0	15.0	15.0	Colombia	Empleado	
	tipo_vivienda	estado_civil	nivel_estudios	genero	\		
SK_CLIENTE							
86570	Propia	Soltero(a)	Profesional	2			
214784	Propia	Union Libre	Profesional	1			
215177	Propia	Union Libre	Especializacion	2			
216798	Familiar	Casado(a)	Profesional	2			
221893	Arrendada	Casado(a)	Especializacion	2			
	no_personas_a_cargo	valor_mvto	cant_productos	Con_Pareja	\		
SK_CLIENTE							
86570	0	118807.0		1		0	
214784	3	85742.0		1		1	
215177	0	231058.0		1		1	
216798	3	571035.0		1		1	
221893	1	463925.0		1		1	
	Educacion	cluster					
SK_CLIENTE							
86570	Profesional	-1					
214784	Profesional	-1					
215177	Postgrado	-1					
216798	Profesional	-1					
221893	Postgrado	-1					