In [1]:

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

In [2]:

```
# Read the CSV file
df_train = pd.read_csv("Train.csv")
```

In [3]:

```
# Display the first few rows of the dataframe
#print(df_train.info())
df_train['fecha_dato'] = pd.to_datetime(df_train['fecha_dato'])
df_train['fecha_alta'] = pd.to_datetime(df_train['fecha_alta'])

df_train['ind_empleado'] = df_train['ind_empleado'].astype('category')
df_train['sexo'] = df_train['sexo'].astype('category')

# Convert pais_residencia to category
df_train['pais_residencia'] = df_train['pais_residencia'].astype('categor

# Convert ind_nuevo to integer
df_train['ind_nuevo'] = df_train['ind_nuevo'].fillna(0).astype('int64')

# Convert ind_nuevo to category
df_train['ind_nuevo'] = df_train['ind_nuevo'].astype('category')

# Convert ult_fec_cli_1t to datetime
df_train['ult_fec_cli_1t'] = pd.to_datetime(df_train['ult_fec_cli_1t'], e

# Convert indrel to category
df_train['indrel'] = df_train['indrel'].astype('category')

# Convert canal_entrada to category
df_train['canal_entrada'] = df_train['canal_entrada'].astype('category')

# Convert tipodom to category
df_train['tipodom'] = df_train['tipodom'].astype('category')

# Step 1: Convert non-numeric values to NaN
df_train['antiguedad'] = pd.to_numeric(df_train['antiguedad'], errors='co

# Drop rows with NaN values in the antiguedad column and convert to int
df_train = df_train.dropna(subset=['antiguedad'])
df_train['antiguedad'] = df_train['antiguedad'].astype('int64')

# Verify the changes
print(df_train['antiguedad'].dtype)
print(df_train['antiguedad'].isnull().sum())
```

```
int64
0
```

In [4]:

```
# Convert to category
df_train['indrel_1mes'] = df_train['indrel_1mes'].astype('category')
df_train['tiprel_1mes'] = df_train['tiprel_1mes'].astype('category')
```

```
df_train['indext'] = df_train['indext'].astype('category')
df_train['conyuemp'] = df_train['conyuemp'].astype('category')
df_train['indfall'] = df_train['indfall'].astype('category')
df_train['cod_prov'] = df_train['cod_prov'].astype('category')
df_train['nomprov'] = df_train['nomprov'].astype('category')
df_train['ind_actividad_cliente'] = df_train['ind_actividad_cliente'].ast
df_train['segmento'] = df_train['segmento'].astype('category')

# Convert age to integer
df_train['age'] = df_train['age'].fillna(0).astype('int64')

# Summary Statistics to ensure data is merged correctly
print(df_train.info())

# Add 0 to the categories of 'conyuemp'
df_train['conyuemp'] = df_train['conyuemp'].astype('category')
df_train['conyuemp'] = df_train['conyuemp'].cat.add_categories([0])

# Substitute NaN values in 'conyuemp' column with 0
df_train['conyuemp'] = df_train['conyuemp'].fillna(0)

# Verify the changes
print(df_train['conyuemp'].isnull().sum())  # Should print 0 if all NaN v

# Check the number of rows and columns
num_rows = df_train.shape[0]
num_columns = df_train.shape[1]
print(f"Number of rows: {num_rows}")
print(f"Number of columns: {num_columns}")

# Remove rows where age is greater than 100
df_train = df_train[df_train['age'] <= 100]

# Add 'Missing' as a new category
df_train['canal_entrada'] = df_train['canal_entrada'].astype('category')
df_train['canal_entrada'] = df_train['canal_entrada'].cat.add_categories(
df_train['canal_entrada'] = df_train['canal_entrada'].fillna('No informat

# Remove the column 'ult_fec_cli_1t' in-place
df_train.drop(columns=['ult_fec_cli_1t'], inplace=True)

# Verify the changes
#print(df_train.columns)

# Drop rows with NaN values in the 'sexo' column
df_train = df_train.dropna(subset=['sexo'])

# Substitute NaN values in 'renta' column with 0 using .loc to avoid Sett
df_train.loc[:, 'renta'] = df_train['renta'].fillna(0)

#get the number of missing data points per column
missing_values_count = df_train.isnull().sum()
#look at the number of missing points in the 48 columns
missing_values_count[0:48]

# Drop rows with NaN values in the 'indrel_1mes' column
df_train = df_train.dropna(subset=['indrel_1mes'])
df_train = df_train.dropna(subset=['tiprel_1mes'])

# Remove all rows with any null values
df_train = df_train.dropna()

# Verify the changes
#print(df_train.isnull().sum())  # Should print 0 for all columns
# print(df_train.shape)  # Check the shape to see how many rows were drop
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 13619575 entries, 0 to 13647308
Data columns (total 48 columns):
 #   Column                 Dtype
---  ------                 -----
 0   fecha_dato             datetime64[ns]
 1   ncodpers               int64
 2   ind_empleado           category
 3   pais_residencia        category
 4   sexo                   category
 5   age                    int64
 6   fecha_alta             datetime64[ns]
 7   ind_nuevo              category
 8   antiguedad             int64
 9   indrel                 category
 10  ult_fec_cli_1t         datetime64[ns]
 11  indrel_1mes            category
 12  tiprel_1mes            category
 13  indresi                object
 14  indext                 category
 15  conyuemp               category
 16  canal_entrada          category
 17  indfall                category
 18  tipodom                category
 19  cod_prov               category
 20  nomprov                category
 21  ind_actividad_cliente  category
 22  renta                  float64
 23  segmento               category
 24  ind_ahor_fin_ult1      int64
 25  ind_aval_fin_ult1      int64
 26  ind_cco_fin_ult1       int64
 27  ind_cder_fin_ult1      int64
 28  ind_cno_fin_ult1       int64
 29  ind_ctju_fin_ult1      int64
 30  ind_ctma_fin_ult1      int64
 31  ind_ctop_fin_ult1      int64
 32  ind_ctpp_fin_ult1      int64
 33  ind_deco_fin_ult1      int64
 34  ind_deme_fin_ult1      int64
 35  ind_dela_fin_ult1      int64
 36  ind_ecue_fin_ult1      int64
 37  ind_fond_fin_ult1      int64
 38  ind_hip_fin_ult1       int64
 39  ind_plan_fin_ult1      int64
 40  ind_pres_fin_ult1      int64
 41  ind_reca_fin_ult1      int64
 42  ind_tjcr_fin_ult1      int64
 43  ind_valo_fin_ult1      int64
 44  ind_viv_fin_ult1       int64
 45  ind_nomina_ult1        float64
 46  ind_nom_pens_ult1      float64
 47  ind_recibo_ult1        int64
dtypes: category(16), datetime64[ns](3), float64(3), int64(25), object(1)
memory usage: 3.6+ GB
None
0
Number of rows: 13619575
Number of columns: 48
```

In [5]:

```
# Display the column names to check for exact matches
#print(df_train.columns)
```

In [6]:

```
# Ensure that the column names to remove are correctly specified
columns_to_remove = ['tipodom', 'Deceased_index', 'Province_code', 'indre

# Drop the specified columns
df_train = df_train.drop(columns=columns_to_remove, errors='ignore')
```

In [7]:

```
# Display the column names to check for exact matches
print(df_train.columns)
```

```
Index(['fecha_dato', 'ncodpers', 'ind_empleado', 'pais_residencia', 'sex
o',
       'age', 'fecha_alta', 'ind_nuevo', 'antiguedad', 'indrel_1mes',
       'tiprel_1mes', 'indresi', 'indext', 'conyuemp', 'canal_entrada',
       'indfall', 'cod_prov', 'nomprov', 'ind_actividad_cliente', 'renta',
       'segmento', 'ind_ahor_fin_ult1', 'ind_aval_fin_ult1',
       'ind_cco_fin_ult1', 'ind_cder_fin_ult1', 'ind_cno_fin_ult1',
       'ind_ctju_fin_ult1', 'ind_ctma_fin_ult1', 'ind_ctop_fin_ult1',
       'ind_ctpp_fin_ult1', 'ind_deco_fin_ult1', 'ind_deme_fin_ult1',
       'ind_dela_fin_ult1', 'ind_ecue_fin_ult1', 'ind_fond_fin_ult1',
       'ind_hip_fin_ult1', 'ind_plan_fin_ult1', 'ind_pres_fin_ult1',
       'ind_reca_fin_ult1', 'ind_tjcr_fin_ult1', 'ind_valo_fin_ult1',
       'ind_viv_fin_ult1', 'ind_nomina_ult1', 'ind_nom_pens_ult1',
       'ind_recibo_ult1'],
      dtype='object')
```

In [8]:

```
# Renaming columns
df_train.rename(columns={'ncodpers': 'Customer_Code'}, inplace=True)
df_train.rename(columns={'ind_empleado': 'Employee_index'}, inplace=True)
df_train.rename(columns={'pais_residencia': 'Country_of_Residence'}, inpl
df_train.rename(columns={'sexo': 'Sex'}, inplace=True)
df_train.rename(columns={'age': 'Age'}, inplace=True)
df_train.rename(columns={'ind_nuevo': 'New_customer_Index'}, inplace=True
df_train.rename(columns={'antiguedad': 'Seniority'}, inplace=True)
df_train.rename(columns={'indrel_1mes': 'Customer_Type_1st_month'}, inpla
df_train.rename(columns={'tiprel_1mes': 'Customer_relation_Type_1st_month
df_train.rename(columns={'indresi': 'Residence_index'}, inplace=True)
df_train.rename(columns={'indext': 'Foreigner_index'}, inplace=True)
df_train.rename(columns={'canal_entrada': 'Channel_used_to_join'}, inplac
df_train.rename(columns={'nomprov': 'Province_name'}, inplace=True)
df_train.rename(columns={'ind_actividad_cliente': 'Activity_index'}, inpl
df_train.rename(columns={'renta': 'Gross_income'}, inplace=True)
df_train.rename(columns={'segmento': 'Segmentation'}, inplace=True)
```

In [9]:

```
df_train.rename(columns={'ind_ahor_fin_ult1': 'Savings_account'}, inplace
df_train.rename(columns={'ind_aval_fin_ult1': 'Guarantees'}, inplace=True
df_train.rename(columns={'ind_cco_fin_ult1': 'Current_accounts'}, inplace
df_train.rename(columns={'ind_cder_fin_ult1': 'Derivada_account'}, inplac
df_train.rename(columns={'ind_cno_fin_ult1': 'Payroll_account'}, inplace=
df_train.rename(columns={'ind_ctju_fin_ult1': 'Junior_account'}, inplace=
df_train.rename(columns={'ind_ctma_fin_ult1': 'Mas_Particular_account'},
df_train.rename(columns={'ind_ctop_fin_ult1': 'Particular_account'}, inpl
df_train.rename(columns={'ind_ctpp_fin_ult1': 'Particular_plus_account'},
df_train.rename(columns={'ind_deco_fin_ult1': 'Short_term_deposits'}, inp
df_train.rename(columns={'ind_deme_fin_ult1': 'Medium_term_deposits'}, in
df_train.rename(columns={'ind_dela_fin_ult1': 'Long_term_deposits'}, inpl
df_train.rename(columns={'ind_ecue_fin_ult1': 'e-accounts'}, inplace=True
df_train.rename(columns={'ind_fond_fin_ult1': 'Funds'}, inplace=True)
df_train.rename(columns={'ind_hip_fin_ult1': 'Mortgage'}, inplace=True)
df_train.rename(columns={'ind_plan_fin_ult1': 'Pensions'}, inplace=True)
df_train.rename(columns={'ind_pres_fin_ult1': 'Loans'}, inplace=True)
df_train.rename(columns={'ind_reca_fin_ult1': 'Taxes'}, inplace=True)
df_train.rename(columns={'ind_tjcr_fin_ult1': 'Credit_card'}, inplace=Tru
df_train.rename(columns={'ind_valo_fin_ult1': 'Securities'}, inplace=True
df_train.rename(columns={'ind_viv_fin_ult1': 'Home_account'}, inplace=Tru
df_train.rename(columns={'ind_nomina_ult1': 'Payroll'}, inplace=True)
df_train.rename(columns={'ind_nom_pens_ult1': 'Pensions'}, inplace=True)
df_train.rename(columns={'ind_recibo_ult1': 'Direct_debit'}, inplace=True
```

In [10]:

```
#Handle negative values in 'Seniority' by setting them to zero
df_train['Seniority'] = df_train['Seniority'].apply(lambda x: 0 if x < 0

# Ensure 'ind_nuevo' contains only 1 or 0
df_train['New_customer_Index'] = pd.to_numeric(df_train['New_customer_Ind
df_train['New_customer_Index'] = df_train['New_customer_Index'].fillna(0)
df_train['New_customer_Index'] = df_train['New_customer_Index'].apply(lam
```

In [11]:

```
# Ensure 'indrel_1mes' contains only 1, 2, 3, 4, or 'P'
valid_values = {'1', '2', '3', '4', 'P'}
df_train['Customer_Type_1st_month'] = df_train['Customer_Type_1st_month']
df_train['Customer_Type_1st_month'] = df_train['Customer_Type_1st_month']
```

In [12]:

```
# Remove rows where age is greater than 100
df_train = df_train[df_train['Age'] <= 100]
```

In [13]:

```
# Drop rows with NaN values in the 'sexo' column
df_train = df_train.dropna(subset=['Sex'])
```

In [14]:

```
# Substitute NaN values in 'renta' column with 0 using .loc to avoid Sett
df_train.loc[:, 'Gross_income'] = df_train['Gross_income'].fillna(0)
```

In [15]:

```
# Remove all rows with any null values
df_train = df_train.dropna()
```

In [16]:

```
# Map
df_train['Employee_index'] = df_train['Employee_index'].map({'N': 'Not em

# Check if the mapping was successful
```

```
print(df_train['Employee_index'].value_counts())
```

```
Employee_index
Not employed    13371115
Ex-Employed         3537
Fillial             2512
Active              2475
Passive               17
Name: count, dtype: int64
```

In [17]:

```
# Map 'V' and 'H' to 'Women' and 'Men'
df_train['Sex'] = df_train['Sex'].map({'V': 'Women', 'H': 'Men'})

# Check if the mapping was successful
print(df_train['Sex'].value_counts())
```

```
Sex
Women    7294847
Men      6084809
Name: count, dtype: int64
```

In [18]:

```
# Map 'V' and 'H' to 'Women' and 'Men'
df_train['Customer_relation_Type_1st_month'] = df_train['Customer_relatio

# Check if the mapping was successful
print(df_train['Customer_relation_Type_1st_month'].value_counts())
```

```
Customer_relation_Type_1st_month
Inactive           7263436
Active             6116100
Former Customer        120
Name: count, dtype: int64
```

In [19]:

```
df_train['log_gross_income'] = np.log(df_train['Gross_income'] + 1)
df_train['sqrt_gross_income'] = np.sqrt(df_train['Gross_income'])
```

In [20]:

```
# Display the updated DataFrame
print(df_train.head())
```

```
  fecha_dato  Customer_Code Employee_index Country_of_Residence    Sex  Ag
e  \
0 2015-01-28        1375586   Not employed                   ES    Men   3
5
1 2015-01-28        1050611   Not employed                   ES  Women   2
3
2 2015-01-28        1050612   Not employed                   ES  Women   2
3
3 2015-01-28        1050613   Not employed                   ES    Men   2
2
4 2015-01-28        1050614   Not employed                   ES  Women   2
3

  fecha_alta  New_customer_Index  Seniority Customer_Type_1st_month  ...
\
0 2015-01-12                   0          6                       P  ...
1 2012-08-10                   0         35                       P  ...
2 2012-08-10                   0         35                       P  ...
3 2012-08-10                   0         35                       P  ...
4 2012-08-10                   0         35                       P  ...

  Loans Taxes Credit_card Securities Home_account Payroll Pensions  \
0     0     0           0          0            0     0.0      0.0
1     0     0           0          0            0     0.0      0.0
2     0     0           0          0            0     0.0      0.0
3     0     0           0          0            0     0.0      0.0
4     0     0           0          0            0     0.0      0.0

  Direct_debit log_gross_income  sqrt_gross_income
0            0        11.376179         295.327107
1            0        10.478688         188.543735
2            0        11.713252         349.541285
3            0        11.693383         346.086030
4            0         0.000000           0.000000

[5 rows x 47 columns]
```

In [21]:

```
from scipy.stats.mstats import winsorize
df_train['winsorized_gross_income'] = winsorize(df_train['Gross_income'],
```

In [22]:

```
# Calculate the first quartile (Q1) and the third quartile (Q3)
Q1 = df_train['Gross_income'].quantile(0.25)
Q3 = df_train['Gross_income'].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Define the lower and upper thresholds
lower_threshold = Q1 - 1.5 * IQR
upper_threshold = Q3 + 1.5 * IQR

# Filter the DataFrame to remove outliers
df_cleaned1 = df_train[(df_train['Gross_income'] >= lower_threshold) & (d

# Display the result (optional)
#print(df_cleaned1)
```

In [23]:

```python
# Calculate the first quartile (Q1) and the third quartile (Q3) for each
Q1 = df_cleaned1.groupby('Sex')['Age'].quantile(0.25)
Q3 = df_cleaned1.groupby('Sex')['Age'].quantile(0.75)

# Calculate the Interquartile Range (IQR) for each group
IQR = Q3 - Q1

# Define the lower and upper thresholds for each group
lower_threshold = Q1 - 1.5 * IQR
upper_threshold = Q3 + 1.5 * IQR

# Filter the DataFrame to remove outliers
def filter_outliers(group):
    lower = lower_threshold[group.name]
    upper = upper_threshold[group.name]
    return group[(group['Age'] >= lower) & (group['Age'] <= upper)]

df_cleaned2 = df_cleaned1.groupby('Sex').apply(filter_outliers).reset_ind
```

In [24]:

```python
# Calculate the first quartile (Q1) and the third quartile (Q3) for each
Q1 = df_cleaned2.groupby('Channel_used_to_join')['Age'].quantile(0.25)
Q3 = df_cleaned2.groupby('Channel_used_to_join')['Age'].quantile(0.75)

# Calculate the Interquartile Range (IQR) for each channel
IQR = Q3 - Q1

# Define the lower and upper thresholds for each channel
lower_threshold = Q1 - 1.5 * IQR
upper_threshold = Q3 + 1.5 * IQR

# Function to filter out outliers based on IQR
def filter_outliers(group):
    lower = lower_threshold[group.name]
    upper = upper_threshold[group.name]
    return group[(group['Age'] >= lower) & (group['Age'] <= upper)]

# Apply the filter function to each group
df_no_outliers = df_cleaned2.groupby('Channel_used_to_join').apply(filter
```

In [26]:

```python
# Specify the filename for the new CSV file
filename = 'df_no_outliers.csv'

# Save the DataFrame to a new CSV file
df_no_outliers.to_csv(filename, index=False)

print(f'DataFrame saved to {filename}')
```

```
DataFrame saved to df_no_outliers.csv
```