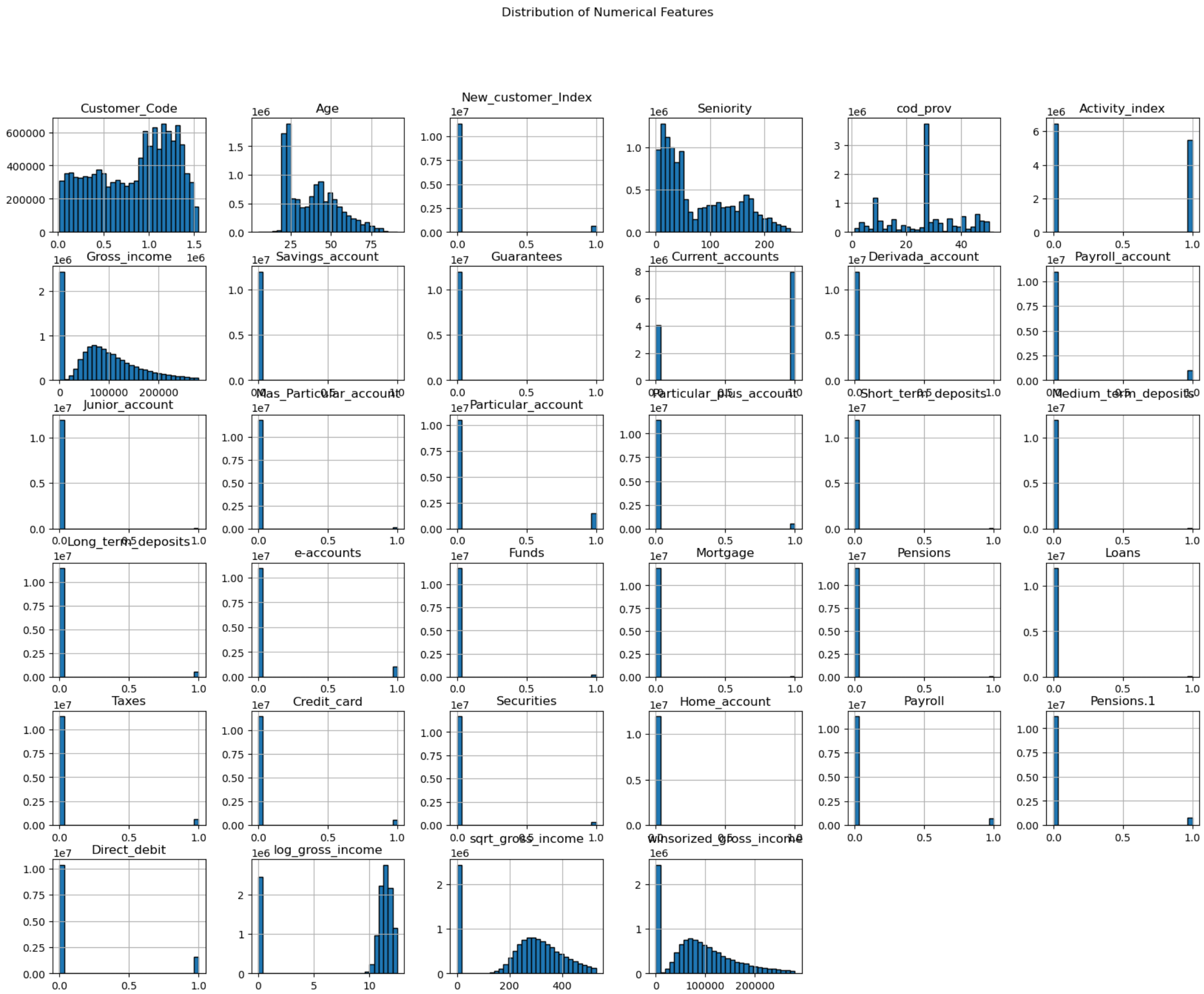
```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
        import matplotlib.pyplot as plt
```

```
In [2]: # Read the CSV file
        df_train = pd.read_csv("df_no_outliers.csv")
```

```
In [3]: # 2 - Univariate Analysis
        # Distribution of numerical features
        df_train.hist(bins=30, figsize=(20, 15), edgecolor='black')
        plt.suptitle('Distribution of Numerical Features')
        plt.show()

        # Summary statistics for categorical columns
        print("Categorical Columns Description:")
        print(df_train.describe(include=['category']))
```



```
Categorical Columns Description:
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call last
)
Cell In[3], line 9
```

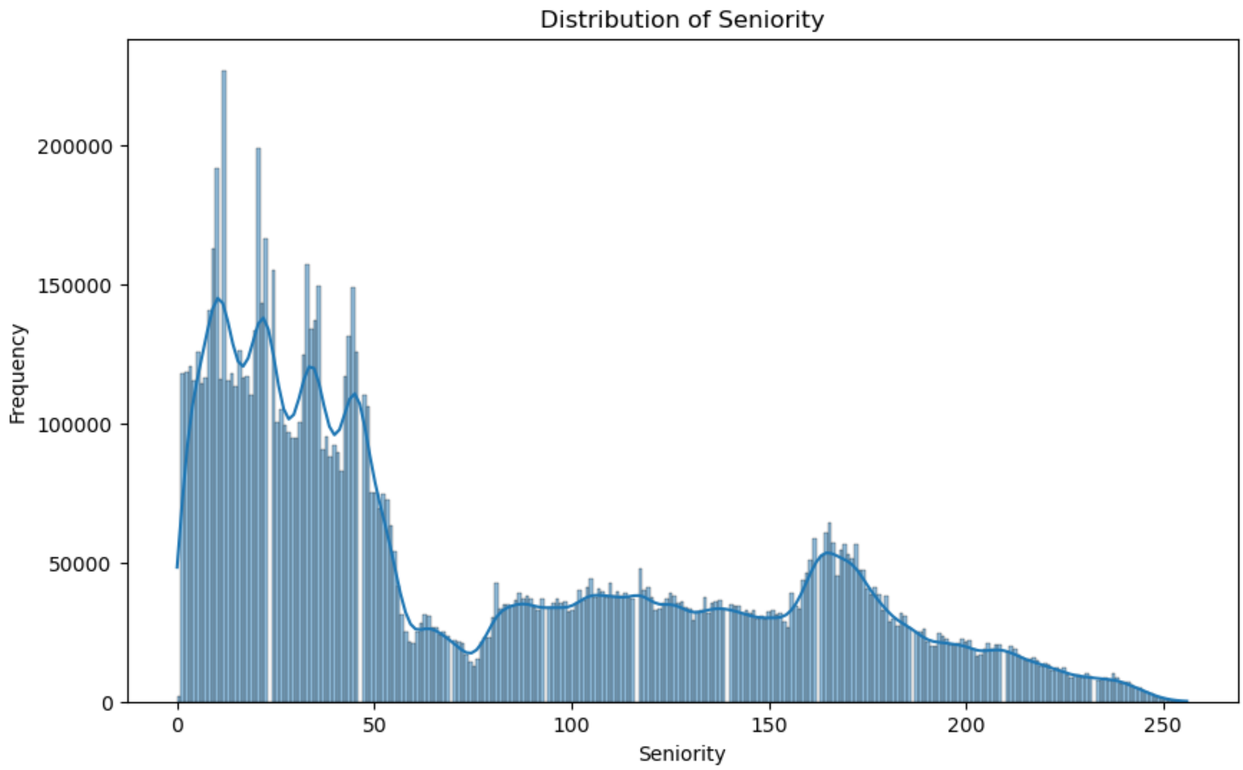
```
      7 # Summary statistics for categorical columns
      8 print("Categorical Columns Description:")
----> 9 print(df_train.describe(include=['category']))

File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/generic.py:11
552, in NDFrame.describe(self, percentiles, include, exclude)
  11310 @final
  11311 def describe(
  11312     self,
   (...)
  11315     exclude=None,
  11316 ) -> Self:
  11317     """
  11318     Generate descriptive statistics.
  11319 
   (...)
  11550     max            NaN      3.0
  11551     """
> 11552     return describe_ndframe(
  11553         obj=self,
  11554         include=include,
  11555         exclude=exclude,
  11556         percentiles=percentiles,
  11557     ).__finalize__(self, method="describe")

File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/methods/descr
ibe.py:97, in describe_ndframe(obj, include, exclude, percentiles)
     90 else:
     91     describer = DataFrameDescriber(
     92         obj=cast("DataFrame", obj),
     93         include=include,
     94         exclude=exclude,
     95     )
---> 97 result = describer.describe(percentiles=percentiles)
     98 return cast(NDFrameT, result)

File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/methods/descr
ibe.py:173, in DataFrameDescriber.describe(self, percentiles)
    170     ldesc.append(describe_func(series, percentiles))
    172 col_names = reorder_columns(ldesc)
--> 173 d = concat(
    174     [x.reindex(col_names, copy=False) for x in ldesc],
    175     axis=1,
    176     sort=False,
    177 )
    178 d.columns = data.columns.copy()
    179 return d

File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/reshape/conca
t.py:380, in concat(objs, axis, join, ignore_index, keys, levels, names, v
erify_integrity, sort, copy)
    377 elif copy and using_copy_on_write():
    378     copy = False
--> 380 op = _Concatenator(
    381     objs,
    382     axis=axis,
    383     ignore_index=ignore_index,
    384     join=join,
    385     keys=keys,
    386     levels=levels,
    387     names=names,
    388     verify_integrity=verify_integrity,
    389     copy=copy,
    390     sort=sort,
    391 )
    393 return op.get_result()

File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/reshape/conca
t.py:443, in _Concatenator.__init__(self, objs, axis, join, keys, levels,
names, ignore_index, verify_integrity, copy, sort)
    440 self.verify_integrity = verify_integrity
    441 self.copy = copy
--> 443 objs, keys = self._clean_keys_and_objs(objs, keys)
    445 # figure out what our result ndim is going to be
    446 ndims = self._get_ndims(objs)

File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/reshape/conca
t.py:505, in _Concatenator._clean_keys_and_objs(self, objs, keys)
    502     objs_list = list(objs)
    504 if len(objs_list) == 0:
--> 505     raise ValueError("No objects to concatenate")
    507 if keys is None:
    508     objs_list = list(com.not_none(*objs_list))

ValueError: No objects to concatenate
```

In [ ]:

```python
# Check if 'Gross_income' column exists
if 'Gross_income' in df_train.columns:
    # Apply logarithmic transformation to the 'Gross_income' column
    log_data = np.log1p(df_train['Gross_income'])

    # Create a histogram for the log-transformed 'Gross_income' column
    plt.figure(figsize=(10, 6))
    sns.histplot(log_data, kde=True)
    plt.title('Log-Transformed Distribution of Gross Income')
    plt.xlabel('Log of Gross Income')
    plt.ylabel('Frequency')
    plt.ylim(0, 500000)  # Set the y-axis limit
    plt.show()
else:
    print("The 'Gross_income' column does not exist in the dataset.")
```

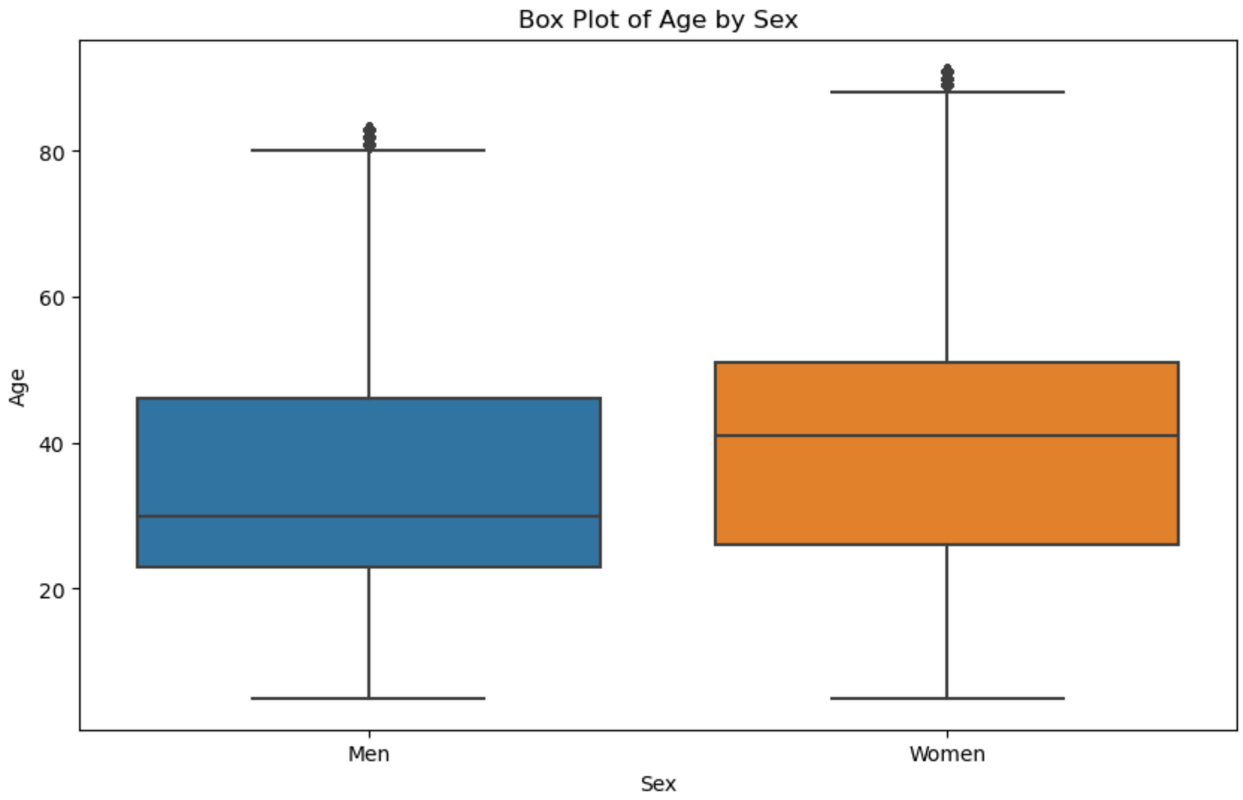
In [20]:

```python
# Check if Seniority column exists
if 'Seniority' in df_train.columns:
    # Create a histogram for the Seniority column
    plt.figure(figsize=(10, 6))
    sns.histplot(df_train['Seniority'], kde=True)
    plt.title('Distribution of Seniority')
    plt.xlabel('Seniority')
    plt.ylabel('Frequency')
    plt.show()
else:
    print("The Seniority column does not exist in the dataset.")
```



In [21]:

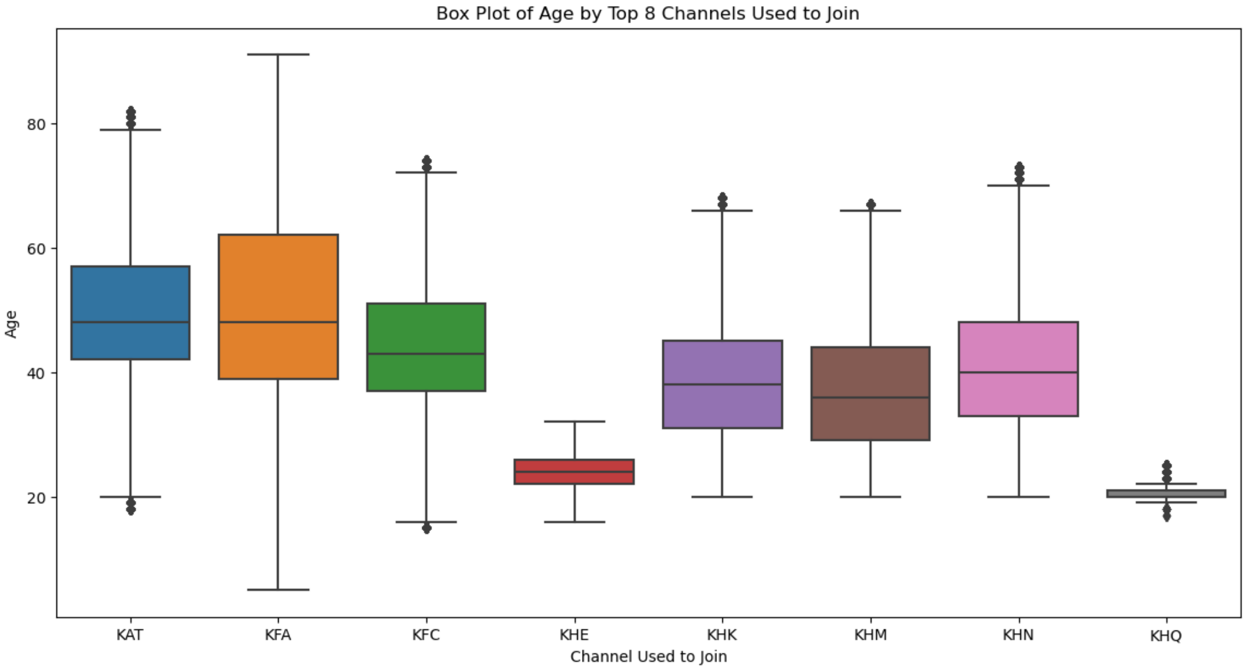
```
# Plot the cleaned data
plt.figure(figsize=(10, 6))
sns.boxplot(x='Sex', y='Age', data=df_train)
plt.title('Box Plot of Age by Sex')
plt.xlabel('Sex')
plt.ylabel('Age')
plt.show()
```



In [22]:

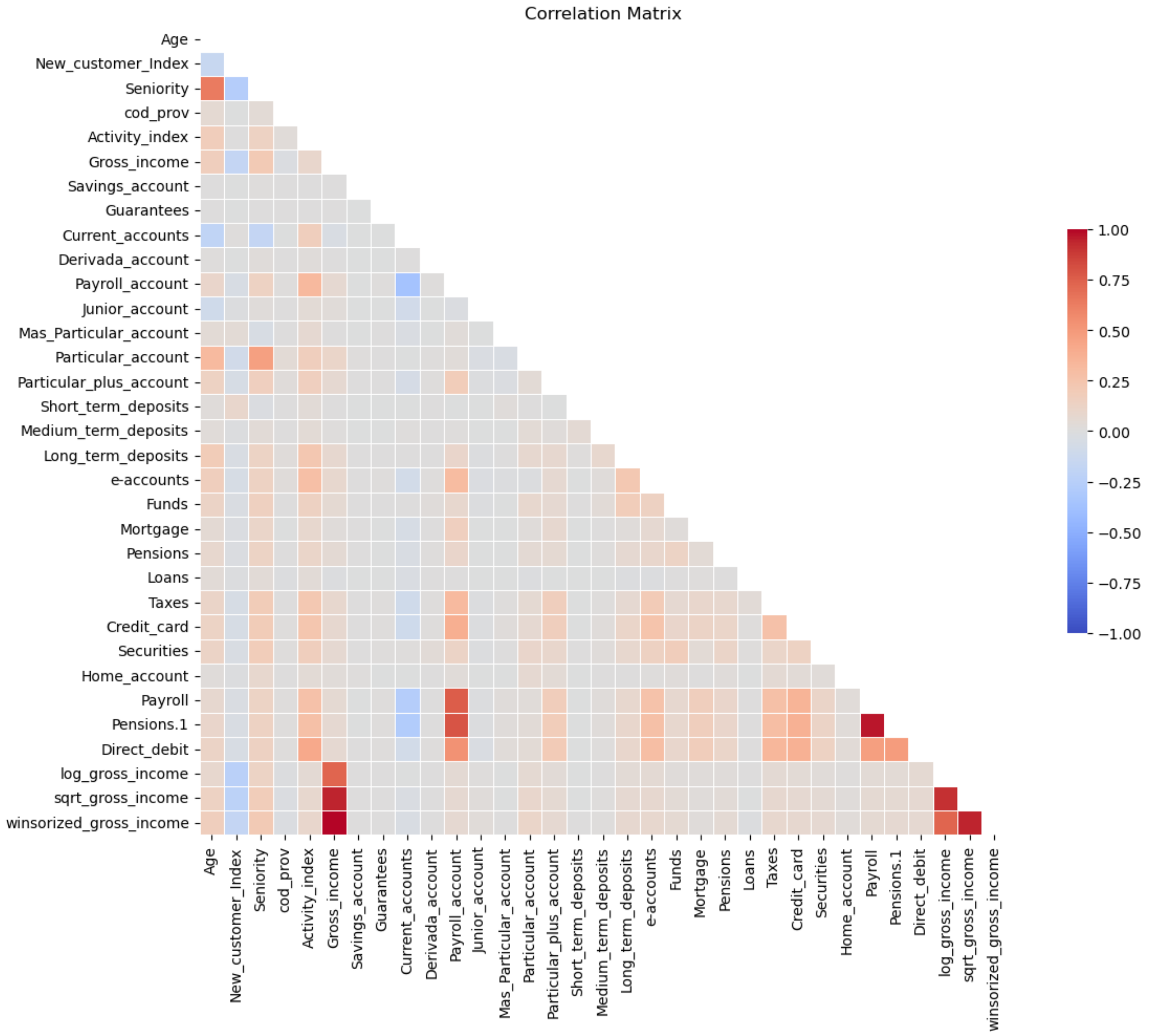
```
# Calculate the count of each channel
channel_counts = df_train['Channel_used_to_join'].value_counts()
```

```
# Get the top 8 channels
top_8_channels = channel_counts.head(8).index

# Filter the DataFrame to include only the top 5 channels
df_top_8_channels = df_train[df_train['Channel_used_to_join'].isin(top_8_

# Create the box plot
plt.figure(figsize=(14, 7))
sns.boxplot(x='Channel_used_to_join', y='Age', data=df_top_8_channels)
plt.title('Box Plot of Age by Top 8 Channels Used to Join')
plt.xlabel('Channel Used to Join')
plt.ylabel('Age')
plt.show()
```



In [23]:

```
# List of columns to exclude from the correlation matrix
columns_to_exclude = ['Customer_Code']  # Add other columns to exclude if

# Drop the specified columns and select numerical columns
relevant_numerical_df = df_train.drop(columns=columns_to_exclude).select_

# Compute the correlation matrix
corr = relevant_numerical_df.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
plt.figure(figsize=(15, 10))

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, annot=True, fmt='.2f', cmap='coolwarm', vmin
            square=True, linewidths=.5, cbar_kws={"shrink": .5})

plt.title('Correlation Matrix')
plt.show()
```



```
In [24]:  # Display the first few rows of the dataframe
          print("First few rows of the dataset:")
          print(df_train.head())

          # Display basic information about the dataframe
          print("\nBasic Information:")
          print(df_train.info())

          # Display summary statistics
          print("\nSummary Statistics:")
          print(df_train.describe())

          # Check for missing values
          print("\nMissing Values:")
          print(df_train.isnull().sum())
```

```
First few rows of the dataset:
   fecha_dato  Customer_Code Employee_index Country_of_Residence  Sex  Age  
\
0  2015-01-28         851959   Not employed                   ES  Men   36
1  2015-02-28         851959   Not employed                   ES  Men   36
2  2015-03-28         851959   Not employed                   ES  Men   36
3  2015-04-28         851959   Not employed                   ES  Men   36
4  2015-05-28         851959   Not employed                   ES  Men   36
```

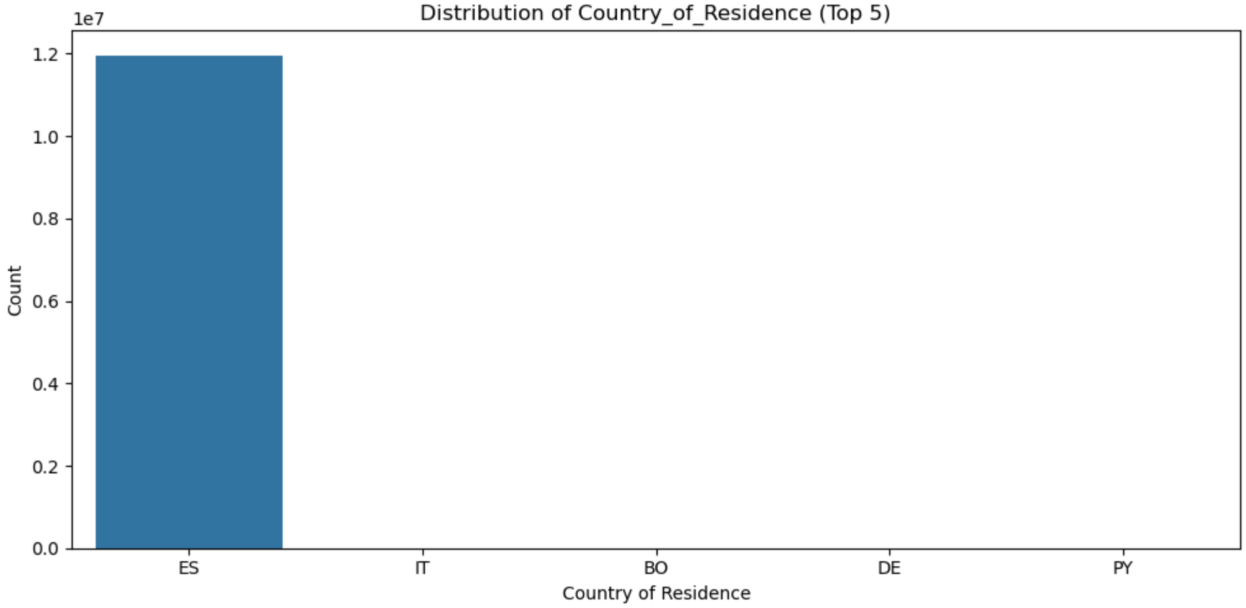
```
   fecha_alta  New_customer_Index  Seniority Customer_Type_1st_month  ...
\
0  2009-09-15                   0         69                       P  ...
1  2009-09-15                   0         69                       P  ...
2  2009-09-15                   0         69                       P  ...
3  2009-09-15                   0         69                       P  ...
4  2009-09-15                   0         69                       P  ...

  Taxes Credit_card Securities Home_account Payroll Pensions.1  Direct_deb
it  \
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

  log_gross_income  sqrt_gross_income  winsorized_gross_income
0          12.0109         405.632321                164537.58
1          12.0109         405.632321                164537.58
2          12.0109         405.632321                164537.58
3          12.0109         405.632321                164537.58
4          12.0109         405.632321                164537.58

[5 rows x 48 columns]

Basic Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11961703 entries, 0 to 11961702
Data columns (total 48 columns):
 #   Column                            Dtype
---  ------                            -----
 0   fecha_dato                        object
 1   Customer_Code                     int64
 2   Employee_index                    object
 3   Country_of_Residence              object
 4   Sex                               object
 5   Age                               int64
 6   fecha_alta                        object
 7   New_customer_Index                int64
 8   Seniority                         int64
 9   Customer_Type_1st_month           object
 10  Customer_relation_Type_1st_month  object
 11  Residence_index                   object
 12  Foreigner_index                   object
 13  conyuemp                          object
 14  Channel_used_to_join              object
 15  indfall                           object
 16  cod_prov                          float64
 17  Province_name                     object
 18  Activity_index                    float64
 19  Gross_income                      float64
 20  Segmentation                      object
 21  Savings_account                  int64
 22  Guarantees                       int64
 23  Current_accounts                 int64
 24  Derivada_account                 int64
 25  Payroll_account                  int64
 26  Junior_account                   int64
 27  Mas_Particular_account           int64
 28  Particular_account               int64
 29  Particular_plus_account          int64
 30  Short_term_deposits              int64
 31  Medium_term_deposits             int64
 32  Long_term_deposits               int64
 33  e-accounts                       int64
 34  Funds                            int64
 35  Mortgage                         int64
 36  Pensions                         int64
 37  Loans                            int64
 38  Taxes                            int64
 39  Credit_card                      int64
 40  Securities                       int64
 41  Home_account                     int64
 42  Payroll                          float64
 43  Pensions.1                       float64
 44  Direct_debit                     int64
 45  log_gross_income                 float64
 46  sqrt_gross_income                float64
 47  winsorized_gross_income          float64
dtypes: float64(8), int64(26), object(14)
memory usage: 4.3+ GB
None

Summary Statistics:
       Customer_Code           Age  New_customer_Index     Seniority  \
count   1.196170e+07  1.196170e+07        1.196170e+07  1.196170e+07   
mean    8.420010e+05  3.877639e+01        5.120843e-02  7.829003e+01   
std     4.265184e+05  1.540047e+01        2.204226e-01  6.574827e+01   
min     1.589000e+04  5.000000e+00        0.000000e+00  0.000000e+00   
25%     4.656960e+05  2.400000e+01        0.000000e+00  2.300000e+01   
50%     9.401310e+05  3.800000e+01        0.000000e+00  4.900000e+01   
75%     1.200944e+06  4.900000e+01        0.000000e+00  1.330000e+02   
max     1.548217e+06  9.100000e+01        1.000000e+00  2.560000e+02   

           cod_prov  Activity_index  Gross_income  Savings_account  \
count  1.196170e+07    1.196170e+07  1.196170e+07     1.196170e+07   
mean   2.661878e+01    4.592029e-01  8.738710e+04     1.001530e-04   
std    1.294053e+01    4.983328e-01  6.695058e+04     1.000714e-02   
min    1.000000e+00    0.000000e+00  0.000000e+00     0.000000e+00   
25%    1.500000e+01    0.000000e+00  4.106738e+04     0.000000e+00   
50%    2.800000e+01    0.000000e+00  8.080068e+04     0.000000e+00   
75%    3.500000e+01    1.000000e+00  1.274842e+05     0.000000e+00   
max    5.200000e+01    1.000000e+00  2.801630e+05     1.000000e+00   

         Guarantees  Current_accounts  ...         Taxes   Credit_card  \
count  1.196170e+07      1.196170e+07  ...  1.196170e+07  1.196170e+07   
mean   2.031483e-05      6.638349e-01  ...  5.211984e-02  4.591127e-02   
std    4.507152e-03      4.723962e-01  ...  2.222687e-01  2.092927e-01   
min    0.000000e+00      0.000000e+00  ...  0.000000e+00  0.000000e+00   
```

```
25%    0.000000e+00      0.000000e+00  ...  0.000000e+00  0.000000e+00
50%    0.000000e+00      1.000000e+00  ...  0.000000e+00  0.000000e+00
75%    0.000000e+00      1.000000e+00  ...  0.000000e+00  0.000000e+00
max    1.000000e+00      1.000000e+00  ...  1.000000e+00  1.000000e+00

         Securities  Home_account       Payroll    Pensions.1  Direct_debi
t  \
count  1.196170e+07  1.196170e+07  1.196170e+07  1.196170e+07  1.196170e+0
7
mean   2.494210e-02  3.895014e-03  5.788791e-02  6.198716e-02  1.333890e-0
1
std    1.559487e-01  6.228839e-02  2.335314e-01  2.411322e-01  3.399947e-0
1
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+0
0
25%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+0
0
50%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+0
0
75%    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+0
0
max    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+0
0

       log_gross_income  sqrt_gross_income  winsorized_gross_income
count      1.196170e+07       1.196170e+07             1.196170e+07
mean       9.133037e+00       2.553196e+02             8.738710e+04
std        4.646379e+00       1.489934e+02             6.695058e+04
min        0.000000e+00       0.000000e+00             0.000000e+00
25%        1.062299e+01       2.026509e+02             4.106738e+04
50%        1.129975e+01       2.842546e+02             8.080068e+04
75%        1.175576e+01       3.570494e+02             1.274842e+05
max        1.254313e+01       5.293043e+02             2.801630e+05

[8 rows x 34 columns]

Missing Values:
fecha_dato                          0
Customer_Code                       0
Employee_index                      0
Country_of_Residence                0
Sex                                 0
Age                                 0
fecha_alta                          0
New_customer_Index                  0
Seniority                           0
Customer_Type_1st_month             0
Customer_relation_Type_1st_month    0
Residence_index                     0
Foreigner_index                     0
conyuemp                            0
Channel_used_to_join                0
indfall                             0
cod_prov                            0
Province_name                       0
Activity_index                      0
Gross_income                        0
```

```
Segmentation                    0
Savings_account                 0
Guarantees                      0
Current_accounts                0
Derivada_account                0
Payroll_account                 0
Junior_account                  0
Mas_Particular_account          0
Particular_account              0
Particular_plus_account         0
Short_term_deposits             0
Medium_term_deposits            0
Long_term_deposits              0
e-accounts                      0
Funds                           0
Mortgage                        0
Pensions                        0
Loans                           0
Taxes                           0
Credit_card                     0
Securities                      0
Home_account                    0
Payroll                         0
Pensions.1                      0
Direct_debit                    0
log_gross_income                0
sqrt_gross_income               0
winsorized_gross_income         0
dtype: int64
```

In [25]:

```
# Assuming df_train is your DataFrame
top_5_countries = df_train['Country_of_Residence'].value_counts().nlarges

# Filter the DataFrame to include only the top 5 countries
df_top_5 = df_train[df_train['Country_of_Residence'].isin(top_5_countries

# Plot the distribution
plt.figure(figsize=(10, 5))
sns.countplot(x='Country_of_Residence', data=df_top_5, order=top_5_countr
plt.title('Distribution of Country_of_Residence (Top 5)')
plt.xlabel('Country of Residence')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

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



```
Country_of_Residence
ES    11961636
IT          17
BO          17
DE          17
PY          16
Name: count, dtype: int64
```

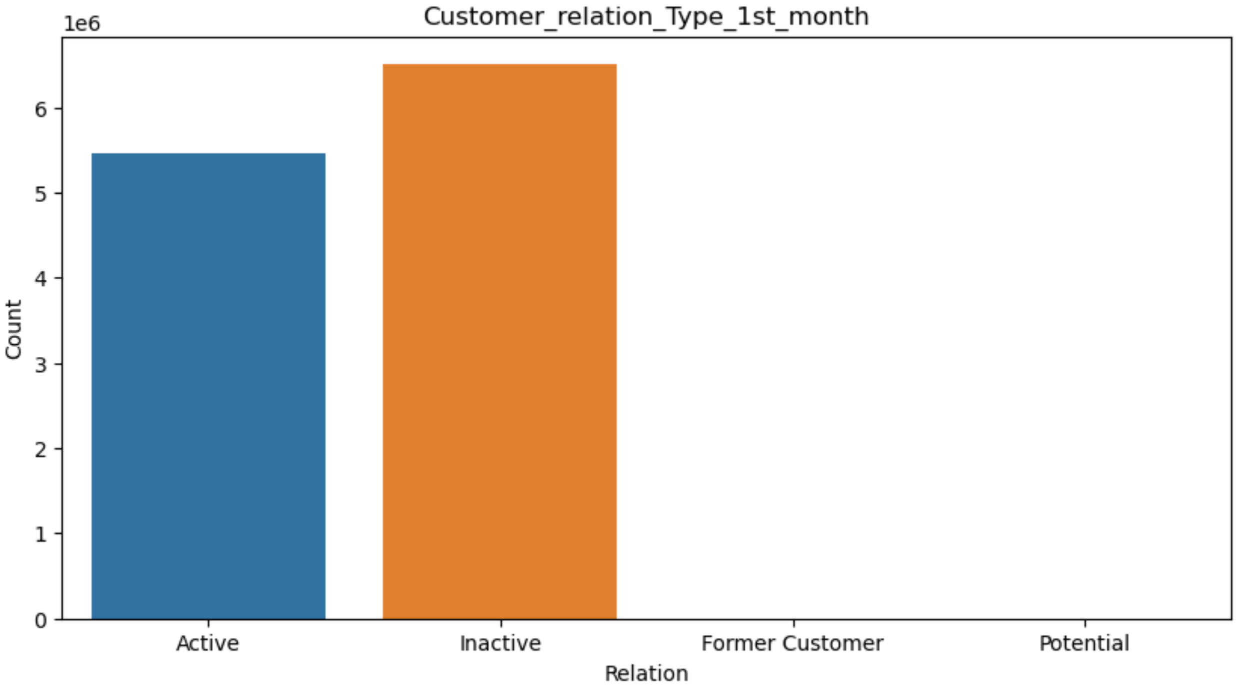
In [26]:

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

# Plot the distribution for the "Sex" column
plt.figure(figsize=(10, 5))
ax = sns.countplot(x='Customer_relation_Type_1st_month', data=df_train, o
plt.title('Customer_relation_Type_1st_month')
plt.xlabel('Relation')
plt.ylabel('Count')
```

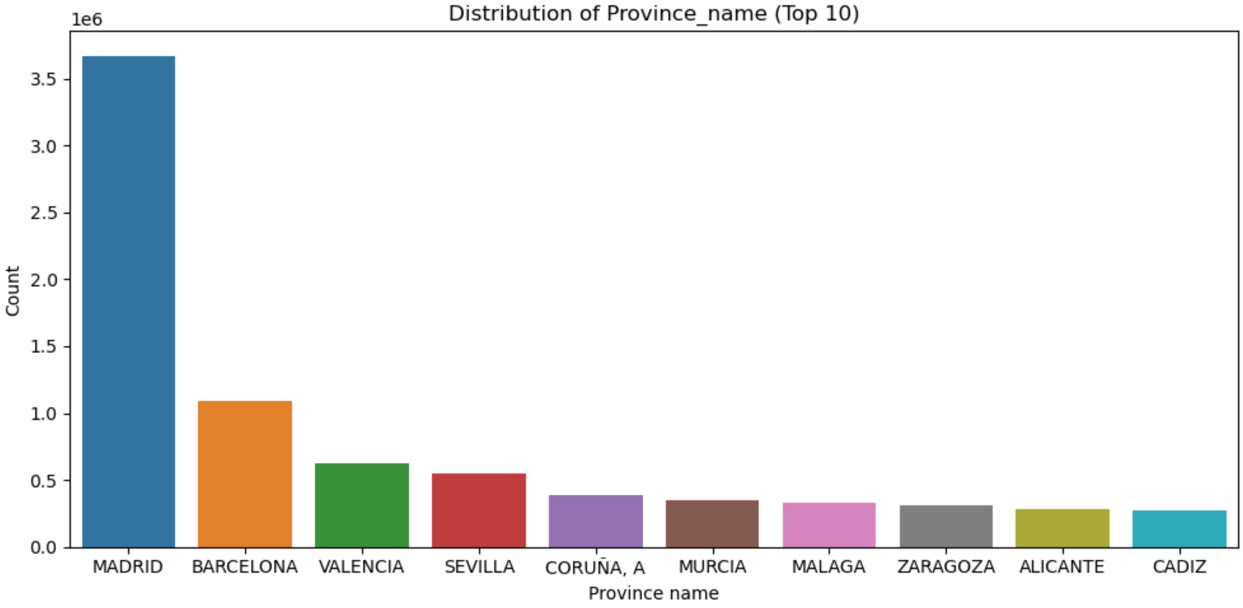
```
Customer_relation_Type_1st_month
Inactive           6504875
Active             5456716
Former Customer        112
Name: count, dtype: int64
```

Out[26]:

```
Text(0, 0.5, 'Count')
```

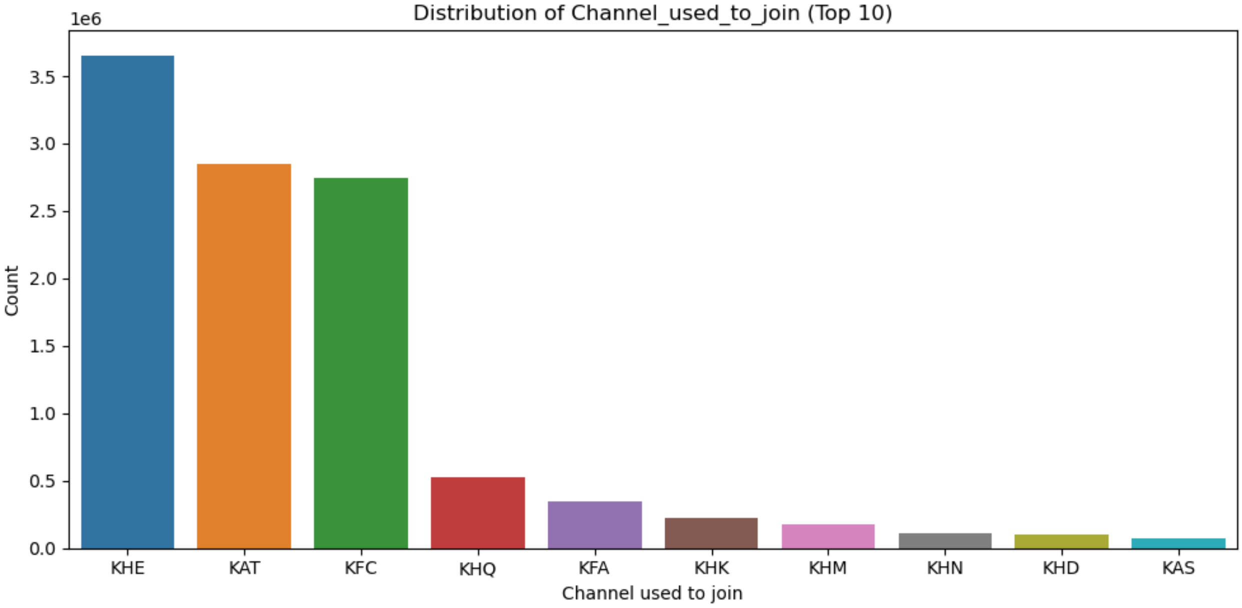


In [27]:
```
# Assuming df_train is your DataFrame
top_10_countries = df_train['Province_name'].value_counts().nlargest(10).

# Filter the DataFrame to include only the top 5 countries
df_top_10 = df_train[df_train['Province_name'].isin(top_10_countries)]

# Plot the distribution
plt.figure(figsize=(10, 5))
sns.countplot(x='Province_name', data=df_top_10, order=top_10_countries)
plt.title('Distribution of Province_name (Top 10)')
plt.xlabel('Province name')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

#print(df_train['Province_name'].value_counts())
```



In [28]:
```
# Assuming df_train is your DataFrame
top_10_countries = df_train['Channel_used_to_join'].value_counts().nlarge
```

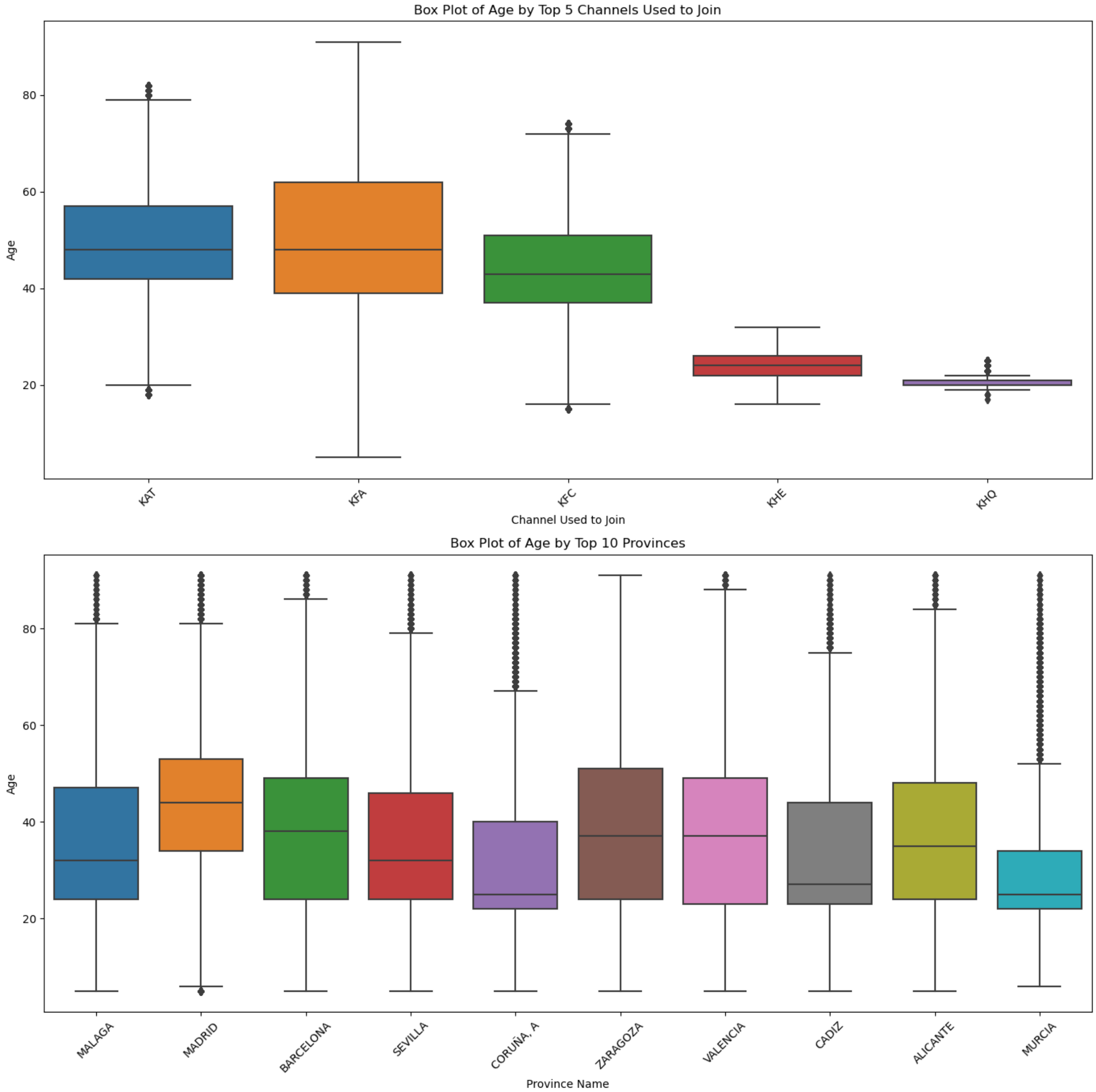
```
# Filter the DataFrame to include only the top 5 countries
df_top_10 = df_train[df_train['Channel_used_to_join'].isin(top_10_countri

# Plot the distribution
plt.figure(figsize=(10, 5))
sns.countplot(x='Channel_used_to_join', data=df_top_10, order=top_10_coun
plt.title('Distribution of Channel_used_to_join (Top 10)')
plt.xlabel('Channel used to join')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



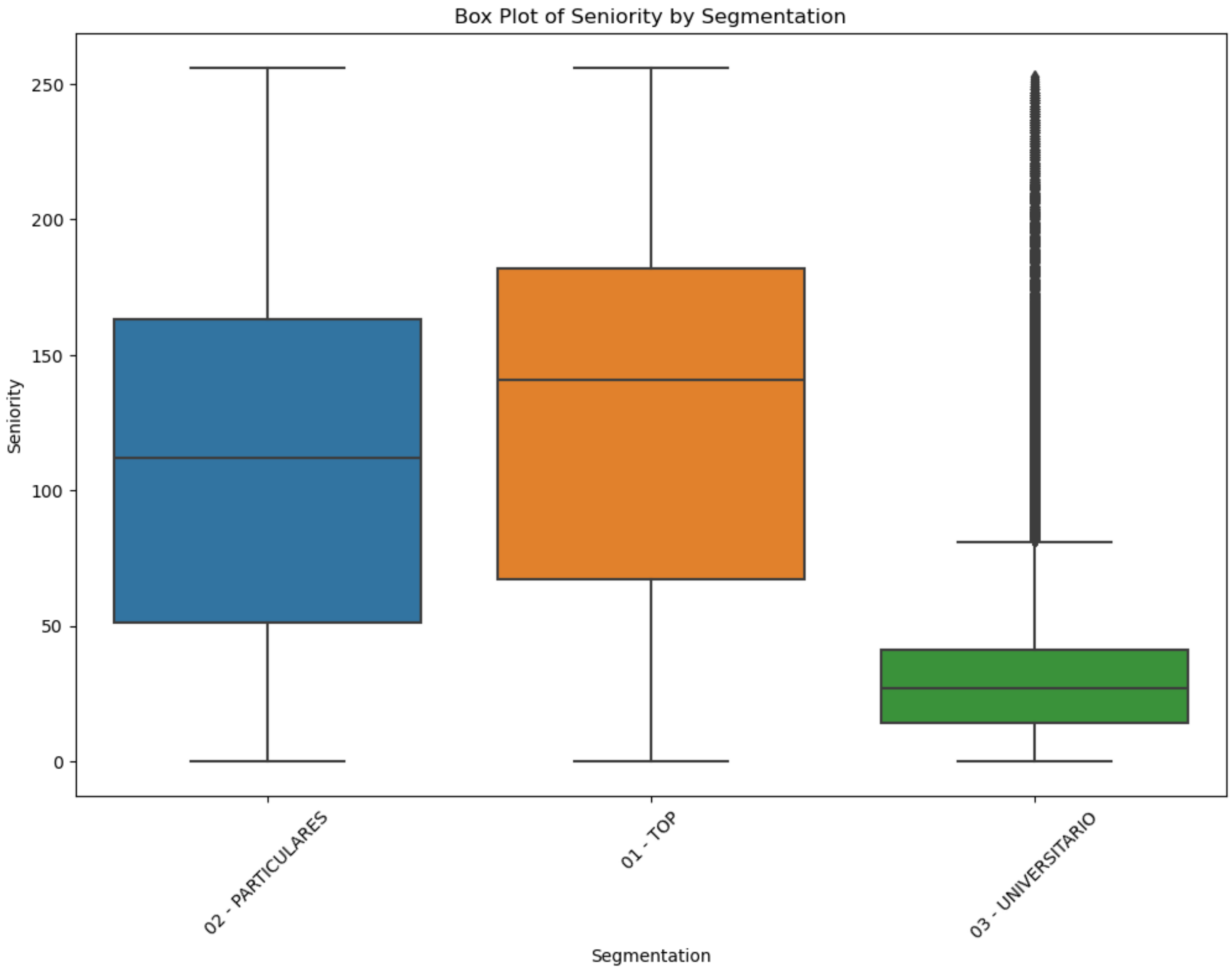
In [29]:

```
# Assuming df_cleaned is your DataFrame containing the relevant data

# Identify the top 5 channels
top_5_channels = df_train['Channel_used_to_join'].value_counts().nlargest

# Filter the DataFrame to include only the top 5 channels
df_top_5_channels = df_train[df_train['Channel_used_to_join'].isin(top_5_

# Identify the top 10 provinces
top_10_provinces = df_train['Province_name'].value_counts().nlargest(10).

# Filter the DataFrame to include only the top 10 provinces
df_top_10_provinces = df_train[df_train['Province_name'].isin(top_10_prov

# Create subplots to compare age distributions
fig, axes = plt.subplots(2, 1, figsize=(14, 14))

# Plot for Top 5 Channels
sns.boxplot(x='Channel_used_to_join', y='Age', data=df_top_5_channels, ax
axes[0].set_title('Box Plot of Age by Top 5 Channels Used to Join')
axes[0].set_xlabel('Channel Used to Join')
axes[0].set_ylabel('Age')

# Plot for Top 10 Provinces
sns.boxplot(x='Province_name', y='Age', data=df_top_10_provinces, ax=axes
axes[1].set_title('Box Plot of Age by Top 10 Provinces')
```

```
axes[1].set_xlabel('Province Name')
axes[1].set_ylabel('Age')

# Rotate x-axis labels for better readability
for ax in axes:
    for label in ax.get_xticklabels():
        label.set_rotation(45)

plt.tight_layout()
plt.show()
```



In [31]:
```
# Check if 'Segmentation' and 'Seniority' columns exist
if 'Segmentation' in df_train.columns and 'Seniority' in df_train.columns
    plt.figure(figsize=(12, 8))
    sns.boxplot(x='Segmentation', y='Seniority', data=df_train)
    plt.title('Box Plot of Seniority by Segmentation')
    plt.xlabel('Segmentation')
    plt.ylabel('Seniority')
    plt.xticks(rotation=45)  # Rotate x-axis labels if necessary
    plt.show()
else:
    print("One or more columns do not exist in the dataset.")
```



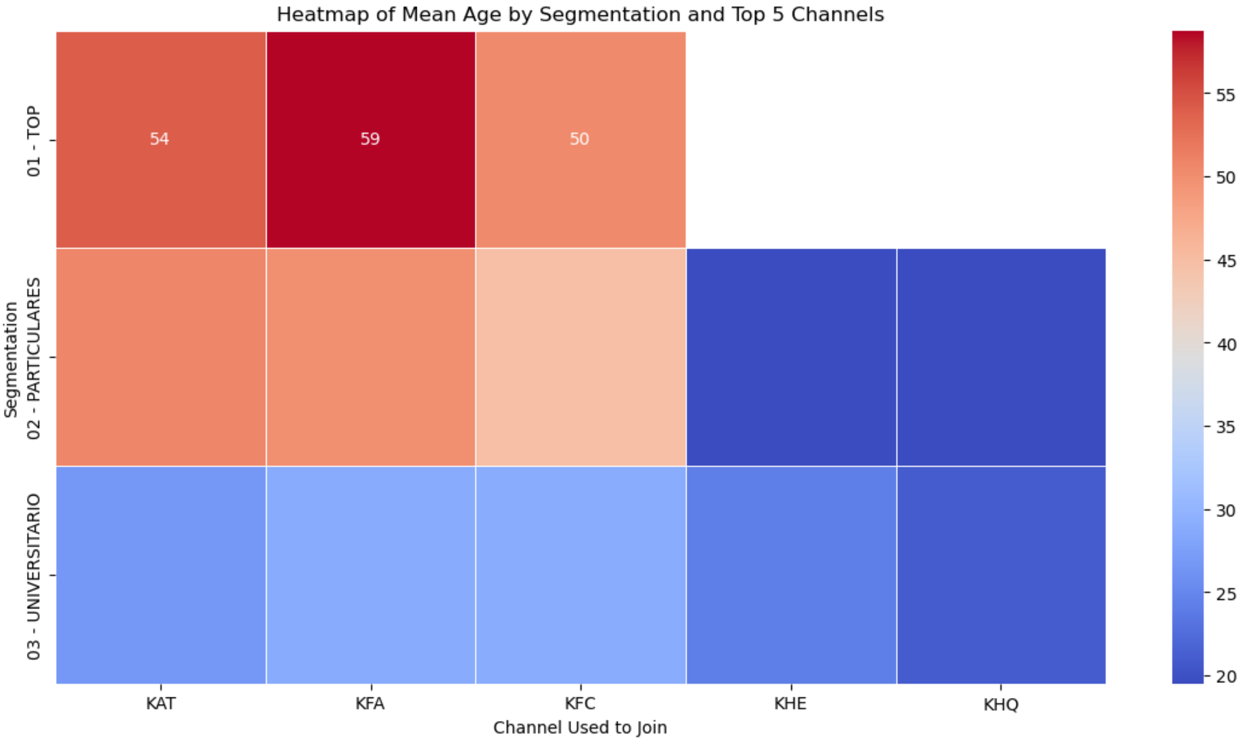
In [ ]:

```
# # Visualize the distribution of numerical features
# numerical_features = df_train.select_dtypes(include=['int64', 'float64'
# # Box plots to visualize the spread and outliers in numerical features
# plt.figure(figsize=(14, 7))
# for i, col in enumerate(numerical_features):
#     plt.subplot(len(numerical_features) // 2 + 1, 2, i + 1)
#     sns.boxplot(y=col, data=df_train)
#     plt.title(f'Box Plot of {col}')
# plt.tight_layout()
# plt.show()
```

In [48]:

```
# Identify the top 5 channels
top_5_channels = df_train['Channel_used_to_join'].value_counts().nlargest

# Filter the DataFrame to include only the top 5 channels
df_top_5_channels = df_train[df_train['Channel_used_to_join'].isin(top_5_

# Create a pivot table for the mean age by segmentation and top 5 channel
pivot_table = df_top_5_channels.pivot_table(values='Age', index='Segmenta

# Print the pivot table for verification
print("\nPivot Table of Mean Age by Segmentation and Top 5 Channels:")
print(pivot_table)

# Create the heatmap
plt.figure(figsize=(14, 7))
sns.heatmap(pivot_table, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Heatmap of Mean Age by Segmentation and Top 5 Channels')
plt.xlabel('Channel Used to Join')
plt.ylabel('Segmentation')
plt.show()
```

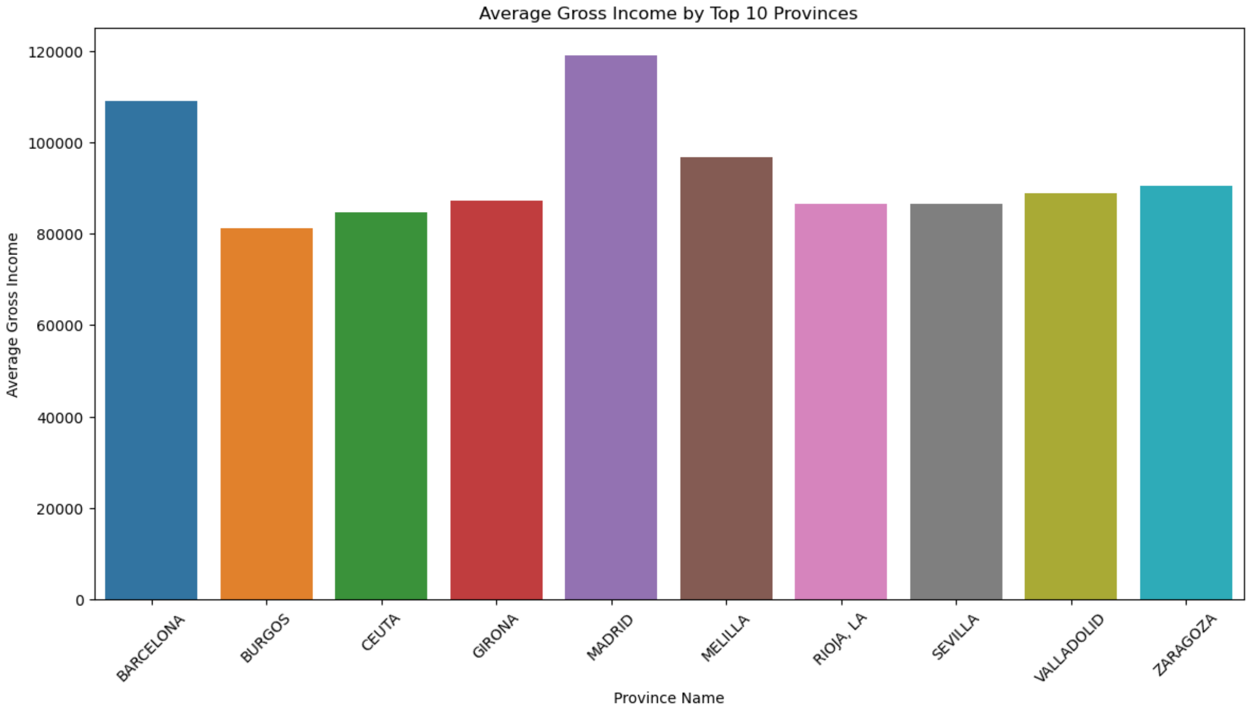
```
Pivot Table of Mean Age by Segmentation and Top 5 Channels:
Channel_used_to_join        KAT        KFA        KFC        KHE        KH
Q
Segmentation
01 - TOP              54.013806  58.764498  50.332913        NaN        Na
N
02 - PARTICULARES     50.785190  49.892479  44.760641  19.601375  19.51855
7
03 - UNIVERSITARIO    26.712131  28.740277  28.970408  24.147726  21.02599
6
```



In [68]:

```
# Calculate the top 10 provinces by average Gross_income
top_10_provinces = df_train.groupby('Province_name')['Gross_income'].mean
print("Top 10 Provinces by Average Gross Income:")
print(top_10_provinces)
# Filter the DataFrame to include only the top 10 provinces
df_top_10_provinces = df_train[df_train['Province_name'].isin(top_10_prov

# Bar plot of average Gross_income by top 10 Province_name
plt.figure(figsize=(14, 7))
avg_income_by_top_10_provinces = df_top_10_provinces.groupby('Province_na
sns.barplot(x='Province_name', y='Gross_income', data=avg_income_by_top_1
plt.title('Average Gross Income by Top 10 Provinces')
plt.xlabel('Province Name')
plt.ylabel('Average Gross Income')
plt.xticks(rotation=45)
plt.show()
```

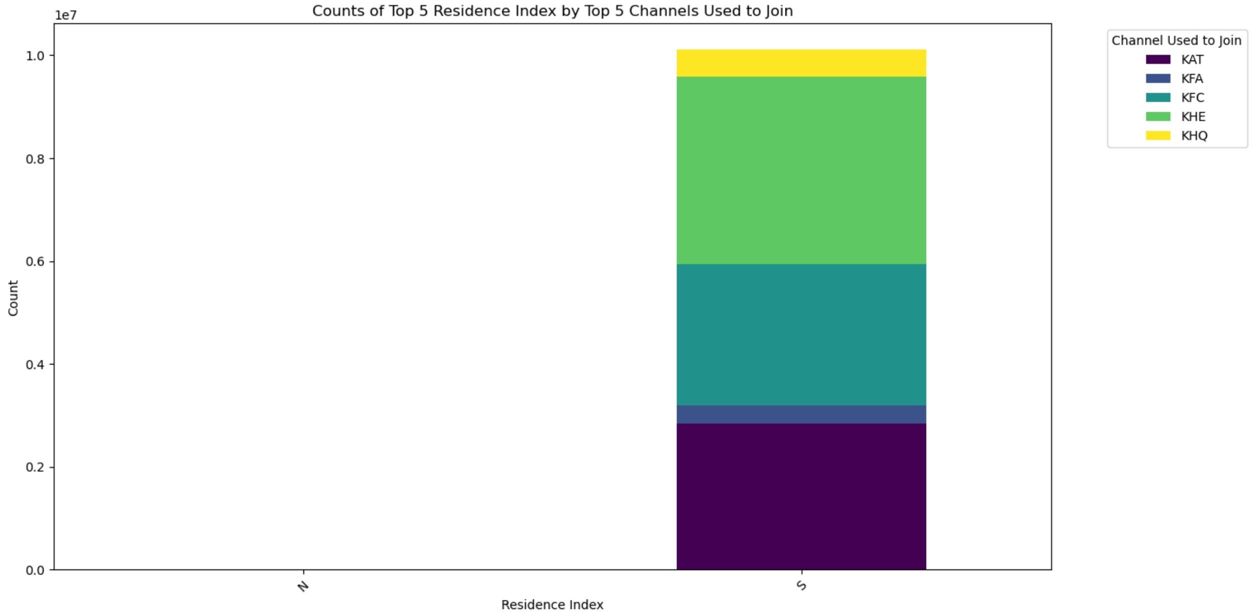
```
Top 10 Provinces by Average Gross Income:
Index(['MADRID', 'BARCELONA', 'MELILLA', 'ZARAGOZA', 'VALLADOLID', 'GIRON
A',
       'SEVILLA', 'RIOJA, LA', 'CEUTA', 'BURGOS'],
      dtype='object', name='Province_name')
```



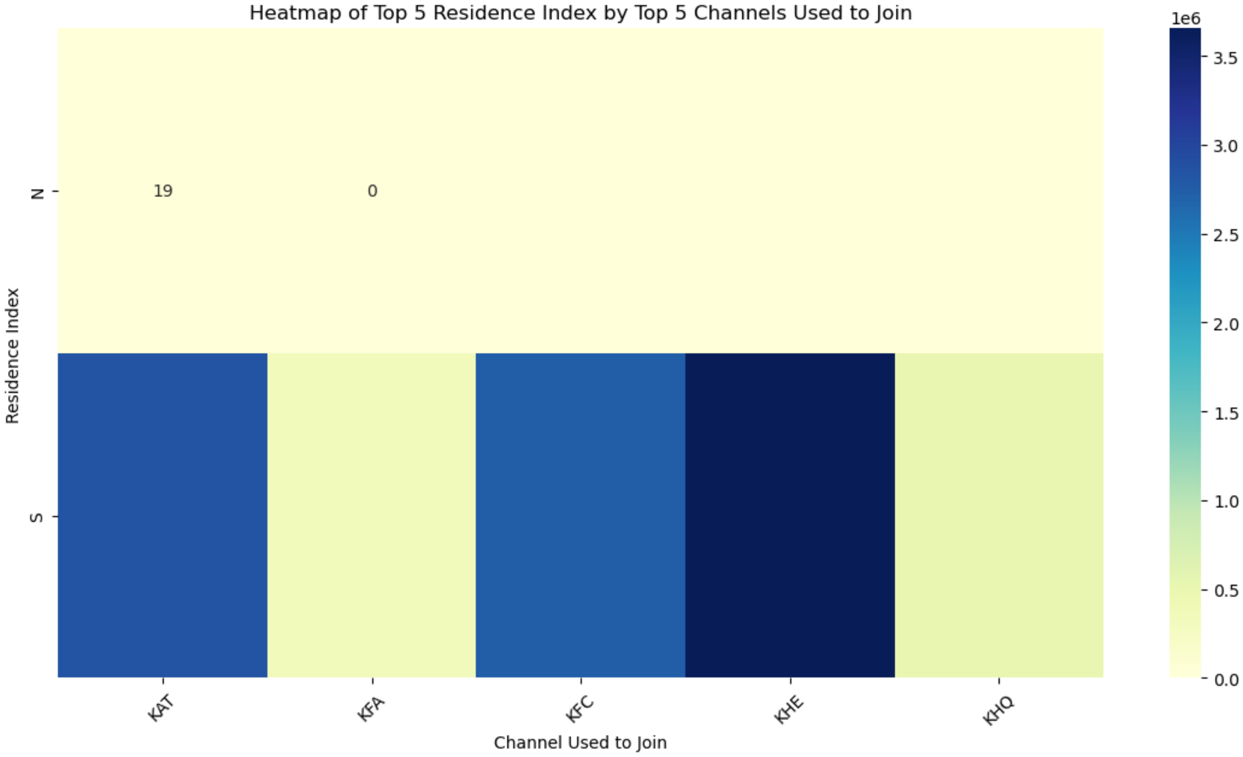
In [93]:

```
# Calculate the top 5 residence_index values by their frequency
top_5_residence_index = df_train['Residence_index'].value_counts().nlarge

# Calculate the top 5 Channel_used_to_join values by their frequency
top_5_channels = df_train['Channel_used_to_join'].value_counts().nlargest

print("Top 5 Residence Index Values:")
print(top_5_residence_index)
print("\nTop 5 Channels Used to Join:")
print(top_5_channels)

# Filter the DataFrame to include only the top 5 residence_index and top
df_top_5 = df_train[df_train['Residence_index'].isin(top_5_residence_inde

# Calculate the counts for each combination of residence_index and Channe
residence_channel_counts_top_5 = df_top_5.groupby(['Residence_index', 'Ch

# Plot the counts using a bar plot
plt.figure(figsize=(14, 7))
residence_channel_counts_top_5.plot(kind='bar', stacked=True, colormap='v
plt.title('Counts of Top 5 Residence Index by Top 5 Channels Used to Join
plt.xlabel('Residence Index')
plt.ylabel('Count')
plt.legend(title='Channel Used to Join', bbox_to_anchor=(1.05, 1), loc='u
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top 5 Residence Index Values:
Index(['S', 'N'], dtype='object', name='Residence_index')

Top 5 Channels Used to Join:
Index(['KHE', 'KAT', 'KFC', 'KHQ', 'KFA'], dtype='object', name='Channel_u
sed_to_join')
<Figure size 1400x700 with 0 Axes>
```


Counts of Top 5 Residence Index by Top 5 Channels Used to Join

```
In [96]: # Plot the counts using a heatmap
         plt.figure(figsize=(14, 7))
         sns.heatmap(residence_channel_counts_top_5, annot=True, fmt='d', cmap='Yl
         plt.title('Heatmap of Top 5 Residence Index by Top 5 Channels Used to Joi
         plt.xlabel('Channel Used to Join')
         plt.ylabel('Residence Index')
         plt.xticks(rotation=45)
         plt.show()
```


Heatmap of Top 5 Residence Index by Top 5 Channels Used to Join