



UNIVERSIDAD EUROPEA DE MADRID

**FACULTAD DE CIENCIAS ECONÓMICAS, EMPRESARIALES Y DE LA
COMUNICACIÓN**

MÁSTER DE FORMACIÓN PERMANENTE EN BUSINESS ANALYTICS

PROYECTO FIN DE MÁSTER - Anexos Técnicos: Notebooks

**DEMANDA HIPOTECARIA EN ESPAÑA: IMPACTO DE
VARIABLES ECONÓMICAS Y PREDICCIÓN MEDIANTE
MODELOS MULTIVARIANTES**

PILAR FERNÁNDEZ LÓPEZ, TOMÁS SANTIAGO ROSALES RIVERA, LUIS FERNÁNDEZ

GUTIÉRREZ

Dirigido por

CARLOS NIETO LÓPEZ VILLALÓN

CURSO 2024-2025

Limpieza de Datos a un nuevo Excel

Como paso posterior a la recopilación de variables en un libro excel (cada variable se ubica en una hoja distinta), tenemos que realizar la limpieza de estas variables. Para ello crearemos otro libro excel con fechas trimestrales desde el 2003 al 2024, asociando a cada trimestre el valor de cada variable.

0. Librerías, Dataset y configuraciones iniciales

```
In [1]: import pandas as pd
import numpy as np
import locale

archivo = "Recolección_variables.xlsx"

# Dataframe con las variables Limpias y ordenadas trimestralmente
df_clean = pd.DataFrame()

# Asignamos estas dos constantes para establecer los límites que vamos a poner para las fechas
__FECHA_INICIO = pd.to_datetime('2003-01-01')
__FECHA_FIN = pd.to_datetime('2023-12-31')
```

```
In [2]: def analizar_nan(df):
    """
    Analiza valores NaN por columna con estadísticas detalladas - función generica
    """
    total_filas = len(df)
    nan_counts = df.isnull().sum()
    nan_percentages = (nan_counts / total_filas * 100).round(2)

    # Crear DataFrame resumen
    resumen = pd.DataFrame({
        'Columna': nan_counts.index,
        'NaN_Count': nan_counts.values,
        'NaN_Percentage': nan_percentages.values,
        'Datos_Válidos': total_filas - nan_counts.values,
        'Total_Filas': total_filas
    })

    # Ordenar por mayor cantidad de NaN
    resumen = resumen.sort_values('NaN_Count', ascending=False)

    return resumen
```

1. Número de hipotecas

```
In [3]: df_1 = pd.read_excel(archivo, sheet_name="NúmeroHipotecas_mes")
df_1.head()
```

	Variable1	Valor1	Variable2	Valor2	Variable3	Valor3	Variable4	Valor4	PERIODO	VALOR
0	Naturaleza de la finca	Viviendas	Concepto financiero	Número de hipotecas	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	2025M02	39.084
1	Naturaleza de la finca	Viviendas	Concepto financiero	Número de hipotecas	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	2025M01	38.058
2	Naturaleza de la finca	Viviendas	Concepto financiero	Número de hipotecas	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	2024M12	32.249
3	Naturaleza de la finca	Viviendas	Concepto financiero	Número de hipotecas	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	2024M11	38.497
4	Naturaleza de la finca	Viviendas	Concepto financiero	Número de hipotecas	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	2024M10	51.535

```
In [4]: # Eliminar las primeras filas con metadatos que no nos interesan
df_1.drop(df_1.columns[[0,1,2,3,4,5,6,7]],axis=1,inplace=True)

#Renombramos las dos columnas
df_1 = df_1.rename(columns={"PERIODO": "Fecha", "VALOR": "Numero Hipotecas"})
# Comprobar el tipo de dato de la columna valor
print("El formato de la columna número de hipotecas es: ",df_1["Numero Hipotecas"].dtype)

df_1.head()
```

El formato de la columna número de hipotecas es: float64

Out[4]:

	Fecha	Numero Hipotecas
0	2025M02	39.084
1	2025M01	38.058
2	2024M12	32.249
3	2024M11	38.497
4	2024M10	51.535

Ajuste de la columna de Fecha

```
In [5]: # Asegurarse de que la columna Fechas es string y eliminar espacios sobrantes
df_1["Fecha"] = df_1["Fecha"].astype(str).str.strip()

# Extrae año y mes
df_1["Año"] = df_1["Fecha"].str[:4].astype(int)
df_1["Mes"] = df_1["Fecha"].str[-2:].astype(int)

# Crea una columna de tipo datetime (día 1 de cada mes)
df_1["Fecha"] = pd.to_datetime(dict(year=df_1["Año"], month=df_1["Mes"], day=1))

# Eliminar columnas temporales
df_1 = df_1.drop(columns=["Año", "Mes"])

df_1.head()
```

Out[5]:

	Fecha	Numero Hipotecas
0	2025-02-01	39.084
1	2025-01-01	38.058
2	2024-12-01	32.249
3	2024-11-01	38.497
4	2024-10-01	51.535

Ajuste de la columna de Numero Hipotecas

Esto lo realizamos por que nos hemos dado cuenta que en Python detecta el punto como separador de miles pero después al pasarlo al excel, según como esté configurado, puede pasarlo como decimales, lo que daría errores. De esta manera nos quitamos esos problemas

```
In [6]: df_1["Numero Hipotecas"] = (df_1["Numero Hipotecas"] * 1000).round().astype(int)
df_1.head()
```

Out[6]:

	Fecha	Numero Hipotecas
0	2025-02-01	39084
1	2025-01-01	38058
2	2024-12-01	32249
3	2024-11-01	38497
4	2024-10-01	51535

Pasar a Trimestral y añadir al dataframe final

```
In [7]: # Ordenamos las fechas
df_1 = df_1.sort_values("Fecha")

# Pasamos a trimestral, en este caso sumando los valores de cada mes
df_1 = df_1.resample("QE", on="Fecha").sum().reset_index()

# Guardamos el DataFrame Limpio en el diccionario
df_clean = df_1

df_clean.head()
```

```
Out[7]:
```

	Fecha	Numero Hipotecas
0	2003-03-31	263600
1	2003-06-30	247071
2	2003-09-30	236977
3	2003-12-31	241791
4	2004-03-31	283170

Establecer los límites superiores e inferiores del DF

```
In [8]: df_clean = df_clean[(df_clean['Fecha'] >= __FECHA_INICIO) & (df_clean['Fecha'] <= __FECHA_FIN)]

print(df_clean.head())
print(len(df_clean))

Fecha Numero Hipotecas
0 2003-03-31 263600
1 2003-06-30 247071
2 2003-09-30 236977
3 2003-12-31 241791
4 2004-03-31 283170
84
```

2. Importe Total Hipotecas

```
In [9]: df_2 = pd.read_excel(archivo, sheet_name="importeHipotecas_mes")
df_2.head()
```

```
Out[9]:
```

	Total Nacional	Unnamed: 1
0	2025M02	6.136.900
1	2025M01	5.793.680
2	2024M12	4.913.994
3	2024M11	5.897.343
4	2024M10	7.758.879

```
In [10]: #Renombramos las columnas
df_2 = df_2.rename(columns={"Total Nacional": "Fecha", "Unnamed: 1": "Importe Hipotecas"})
# Comprobar el tipo de dato de la columna Importe Hipotecas
print("El formato de la columna Importe Hipotecas es: ", df_2["Importe Hipotecas"].dtype)

df_2.head()
```

El formato de la columna Importe Hipotecas es: object

```
Out[10]:
```

	Fecha	Importe Hipotecas
0	2025M02	6.136.900
1	2025M01	5.793.680
2	2024M12	4.913.994
3	2024M11	5.897.343
4	2024M10	7.758.879

Ajuste de la columna de Importe Hipotecas

```
In [11]: # Pasar la columna Importe Hipotecas de object a float
# Viene dato de la forma 6.000.000 y está en miles de euros
df_2["Importe Hipotecas"] = df_2["Importe Hipotecas"].str.replace(".", "", regex=False).astype(float) * 1000
```

Ajuste de la columna de Fecha

```
In [12]: # Asegurarse de que las columnas son strings y eliminar espacios sobrantes
df_2["Fecha"] = df_2["Fecha"].astype(str).str.strip()
# Extrae año y mes
df_2["Año"] = df_2["Fecha"].str[:4].astype(int)
df_2["Mes"] = df_2["Fecha"].str[-2:].astype(int)

# Crea una columna de tipo datetime (día 1 de cada mes)
df_2["Fecha"] = pd.to_datetime(dict(year=df_2["Año"], month=df_2["Mes"], day=1))

# Eliminar columnas temporales
df_2 = df_2.drop(columns=["Año", "Mes"])

df_2.head()
```

```
Out[12]:
```

	Fecha	Importe Hipotecas
0	2025-02-01	6.136900e+09
1	2025-01-01	5.793680e+09
2	2024-12-01	4.913994e+09
3	2024-11-01	5.897343e+09
4	2024-10-01	7.758879e+09

Pasar a Trimestral y ajustar límites

```
In [13]: # Ordenar por fecha
df_2 = df_2.sort_values("Fecha")

# Pasar a trimestral (suma en este caso)
df_2 = df_2.resample("QE", on="Fecha").sum().reset_index()

df_2 = df_2[(df_2['Fecha'] >= __FECHA_INICIO) & (df_2['Fecha'] <= __FECHA_FIN)]
```

Añadir al dataframe final

```
In [14]: df_clean = df_clean.merge(df_2, on="Fecha", how="outer")

df_clean.head()
```

```
Out[14]:
```

	Fecha	Numero Hipotecas	Importe Hipotecas
0	2003-03-31	263600	2.417559e+10
1	2003-06-30	247071	2.340044e+10
2	2003-09-30	236977	2.397833e+10
3	2003-12-31	241791	2.462120e+10
4	2004-03-31	283170	3.002192e+10

3. Valor Medio / Mes

```
In [15]: # En este caso, vamos a calcular esta variable como Importe Total Hipotecas / Número de hipotecas
# Unimos los dos DataFrames por la columna 'Fecha'
df_3 = pd.merge(df_1, df_2, on="Fecha", how="inner")
# Creamos la nueva columna calculada
df_3["valor_medio_hipotecas"] = (df_3["Importe Hipotecas"] / df_3["Número Hipotecas"]).round(2)
df_3 = df_3.drop(columns=["Importe Hipotecas", "Número Hipotecas"])
df_3.head(10)
```

```
Out[15]:
```

	Fecha	valor_medio_hipotecas
0	2003-03-31	91713.17
1	2003-06-30	94711.40
2	2003-09-30	101184.20
3	2003-12-31	101828.44
4	2004-03-31	106020.83
5	2004-06-30	107479.23
6	2004-09-30	114014.39
7	2004-12-31	113890.31
8	2005-03-31	119738.46
9	2005-06-30	121246.85

Añadir al dataframe final

```
In [16]: df_clean = df_clean.merge(df_3, on="Fecha", how="outer")  
df_clean.head()
```

```
Out[16]:
```

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas
0	2003-03-31	263600	2.417559e+10	91713.17
1	2003-06-30	247071	2.340044e+10	94711.40
2	2003-09-30	236977	2.397833e+10	101184.20
3	2003-12-31	241791	2.462120e+10	101828.44
4	2004-03-31	283170	3.002192e+10	106020.83

4. Tipo interés Medio / Mes

```
In [17]: df_4 = pd.read_excel(archivo, sheet_name="tipoInteresMedio_mes")  
df_4.head()
```

	Variable1	Valor1	Variable2	Valor2	Variable3	Valor3	Variable4	Valor4	Variable5	Valor5	PERIODO	VALOR
0	Naturaleza de la finca	Viviendas	Concepto financiero	Tipo de interés medio	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	Tipo de interés	Total	2025M02	2,96
1	Naturaleza de la finca	Viviendas	Concepto financiero	Tipo de interés medio	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	Tipo de interés	Total	2025M01	3,08
2	Naturaleza de la finca	Viviendas	Concepto financiero	Tipo de interés medio	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	Tipo de interés	Total	2024M12	3,25
3	Naturaleza de la finca	Viviendas	Concepto financiero	Tipo de interés medio	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	Tipo de interés	Total	2024M11	3,28
4	Naturaleza de la finca	Viviendas	Concepto financiero	Tipo de interés medio	Total Nacional	Total Nacional	Cambios de conceptos	Base nueva. Mensual	Tipo de interés	Total	2024M10	3,12

```
In [18]: # Eliminar las primeras filas con metadatos que no nos interesan  
df_4.drop(df_4.columns[[0,1,2,3,4,5,6,7,8,9]],axis=1,inplace=True)
```

```
#Renombramos las columnas  
df_4 = df_4.rename(columns={"PERIODO": "Fecha", "VALOR": "Tipo interes Medio"})
```

```
# Comprobar el tipo de dato de la columna Tipo interes Medio
print("El formato de la columna Tipo interes Medio es: ", df_4[ "Tipo interes Medio" ].dtype)
```

```
df_4.head()
```

El formato de la columna Tipo interes Medio es: object

Out[18]:

	Fecha	Tipo interes Medio
0	2025M02	2,96
1	2025M01	3,08
2	2024M12	3,25
3	2024M11	3,28
4	2024M10	3,12

Pasar a Trimestral y ajustar límites

```
In [21]: # Ordenar por fecha
df_4 = df_4.sort_values("Fecha")

# Pasar a trimestral (media en este caso)
df_4 = df_4.resample("QE", on="Fecha").mean().reset_index()

df_4 = df_4[(df_4['Fecha'] >= __FECHA_INICIO) & (df_4['Fecha'] <= __FECHA_FIN)]
```

Añadir al dataframe final

```
In [22]: df_clean = df_clean.merge(df_4, on="Fecha", how="outer")  
        df_clean.head()
```

Out[22]:	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000

5. Tasa de paro

```
In [23]: df_5 = pd.read_excel(archivo, sheet_name="TasaParo_trim")
df_5.head(5)
```

Out[23]:	Comunidades y Ciudades Autónomas	Sexo	Edad	Periodo	Total
0	Total Nacional	Ambos sexos	Total	2025T1	11,36
1	Total Nacional	Ambos sexos	Total	2024T4	10,61
2	Total Nacional	Ambos sexos	Total	2024T3	11,21
3	Total Nacional	Ambos sexos	Total	2024T2	11,27
4	Total Nacional	Ambos sexos	Total	2024T1	12,29

```
In [24]: # Eliminar las primeras filas con metadatos que no nos interesan
df_5.drop(df_5.columns[[0,1,2]],axis=1,inplace=True)

#Renombramos las columnas
df_5 = df_5.rename(columns={"Periodo": "Fecha", "Total": "Tasa Paro (%)"})

df_5.head(5)
```

Out[24]:	Fecha	Tasa Paro (%)
0	2025T1	11,36
1	2024T4	10,61
2	2024T3	11,21
3	2024T2	11,27
4	2024T1	12,29

```
In [25]: print(df_5.dtypes)
```

```
Fecha          object
Tasa Paro (%)  object
dtype: object
```

```
In [26]: df_5["Tasa Paro (%)] = ( df_5["Tasa Paro (%)] .astype(str) # Asegura que todo es string .str.strip() # Elimina espacios al inicio/final
```

```

        .str.replace("\u202f", "", regex=True) # Elimina espacios finos (INE los usa a veces)
        .str.replace(",", ".", regex=False)    # Reemplaza coma por punto
    )

# Ahora convertir a float
df_5["Tasa Paro (%)"] = pd.to_numeric(df_5["Tasa Paro (%)"], errors="coerce")

print(df_5.dtypes)
print(df_5.head())

```

```

Fecha          object
Tasa Paro (%) float64
dtype: object
   Fecha  Tasa Paro (%)
0  2025T1      11.36
1  2024T4      10.61
2  2024T3      11.21
3  2024T2      11.27
4  2024T1      12.29

```

```

In [27]: df_5["Fecha"] = df_5["Fecha"].astype(str).str.strip()

df_5["Año"] = df_5["Fecha"].str[:4].astype(int)
df_5["Trimestre"] = df_5["Fecha"].str[-1:].astype(int)

df_5["Mes"] = df_5["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_5["Fecha"] = pd.to_datetime(dict(year=df_5["Año"], month=df_5["Mes"], day=1))
df_5["Fecha"] = df_5["Fecha"] + pd.offsets.QuarterEnd(0)

df_5 = df_5.drop(columns=["Año", "Trimestre", "Mes"])

df_5.head()

```

```

Out[27]:    Fecha  Tasa Paro (%)
0  2025-03-31      11.36
1  2024-12-31      10.61
2  2024-09-30      11.21
3  2024-06-30      11.27
4  2024-03-31      12.29

```

```

In [28]: df_5 = df_5[(df_5['Fecha'] >= __FECHA_INICIO) & (df_5['Fecha'] <= __FECHA_FIN)]

print(df_5.head())
print(len(df_5))

```

```

Fecha  Tasa Paro (%)
5 2023-12-31      11.80
6 2023-09-30      11.89
7 2023-06-30      11.67
8 2023-03-31      13.38
9 2022-12-31      12.99
84

```

```

In [29]: df_clean = df_clean.merge(df_5, on="Fecha", how="outer")

df_clean.head()

```

```

Out[29]:    Fecha  Numero Hipotecas  Importe Hipotecas  valor_medio_hipotecas  Tipo interes Medio  Tasa Paro (%)
0  2003-03-31      263600      2.417559e+10      91713.17      4.693333      11.99
1  2003-06-30      247071      2.340044e+10      94711.40      4.453333      11.28
2  2003-09-30      236977      2.397833e+10      101184.20      4.213333      11.30
3  2003-12-31      241791      2.462120e+10      101828.44      3.883333      11.37
4  2004-03-31      283170      3.002192e+10      106020.83      3.760000      11.50

```

6. Tipo Contrato

Este conjunto de datos divide la cantidad de contratos de dos tipos, de tipo temporal o de tipo fijo, vamos a crear dos columnas a partir de la información de este conjunto de datos, para finalmente quedarnos por un lado con la cantidad de

contratos temporales y la cantidad de contratos indefinidos.

```
In [30]: df_6 = pd.read_excel(archivo, sheet_name="TipoContrato_trim")  
df_6.drop(df_6.columns[[0,1,2]],axis=1,inplace=True)  
print(df_6.head(5))
```

	Tipo de contrato o relación laboral	Periodo	Total
0	De duración indefinida: Total	2025T1	15.702,70
1	De duración indefinida: Total	2024T4	15.712,50
2	De duración indefinida: Total	2024T3	15.601,50
3	De duración indefinida: Total	2024T2	15.498,80
4	De duración indefinida: Total	2024T1	15.226,80

Mapeo de los resultados para que sean dos columnas calculadas

```
In [31]: #print(f"Valores únicos: {list(df_6["Tipo de contrato o relación laboral"].unique())}")
```

```
In [32]: df_6_contratos = pd.DataFrame()  
df_6_contratos["Periodo"] = df_6["Periodo"].unique()  
df_6_contratos = df_6_contratos.sort_values("Periodo").reset_index(drop=True)  
  
# Asignar los valores de cada tipo de contrato  
df_6_contratos["Contratos indefinidos"] = df_6_contratos["Periodo"].map(  
    df_6[df_6["Tipo de contrato o relación laboral"].str.contains("indefinida", case=False)]\br/>    .set_index("Periodo")["Total"]  
)  
df_6_contratos["Contratos temporales"] = df_6_contratos["Periodo"].map(  
    df_6[df_6["Tipo de contrato o relación laboral"].str.contains("Temporal", case=False)]\br/>    .set_index("Periodo")["Total"]  
)  
  
print(df_6_contratos.head())  
print(len(df_6_contratos))
```

	Periodo	Contratos indefinidos	Contratos temporales
0	2002T1	9.039,20	4.264,30
1	2002T2	9.209,30	4.375,70
2	2002T3	9.374,70	4.402,80
3	2002T4	9.453,20	4.416,40
4	2003T1	9.568,00	4.370,20

Convertir las columnas a float

```
In [33]: df_6 = df_6_contratos  
print(df_6.dtypes)
```

	Periodo	Contratos indefinidos	Contratos temporales
	object	object	object
	object	object	object
	object	object	object

```
In [34]: for col in ["Contratos indefinidos", "Contratos temporales"]:  
    df_6[col] = (  
        df_6[col]  
        .astype(str)  
        .str.strip()  
        .str.replace("\u202f", "", regex=True)  
        .str.replace(".", "", regex=False)  
        .str.replace(",", ".", regex=False)  
    )  
    df_6[col] = pd.to_numeric(df_6[col], errors="coerce")  
  
print(df_6.dtypes)  
print(df_6.head())
```

```

Periodo          object
Contratos indefinidos  float64
Contratos temporales  float64
dtype: object
   Periodo  Contratos indefinidos  Contratos temporales
0  2002T1           9039.2           4264.3
1  2002T2           9209.3           4375.7
2  2002T3           9374.7           4402.8
3  2002T4           9453.2           4416.4
4  2003T1           9568.0           4370.2

```

Renombra la columna Periodo a Fecha y cambiar a tipo datetime

```

In [35]: df_6 = df_6.rename(columns={"Periodo": "Fecha"})

df_6["Fecha"] = df_6["Fecha"].astype(str).str.strip()

df_6["Año"] = df_6["Fecha"].str[:4].astype(int)
df_6["Trimestre"] = df_6["Fecha"].str[-1:].astype(int)

df_6["Mes"] = df_6["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_6["Fecha"] = pd.to_datetime(dict(year=df_6["Año"], month=df_6["Mes"], day=1))
df_6["Fecha"] = df_6["Fecha"] + pd.offsets.QuarterEnd(0)

df_6 = df_6.drop(columns=["Año", "Trimestre", "Mes"])

df_6.head()

```

```

Out[35]:    Fecha  Contratos indefinidos  Contratos temporales
0  2002-03-31           9039.2           4264.3
1  2002-06-30           9209.3           4375.7
2  2002-09-30           9374.7           4402.8
3  2002-12-31           9453.2           4416.4
4  2003-03-31           9568.0           4370.2

```

Establecer los límites

```

In [36]: df_6 = df_6[(df_6['Fecha'] >= __FECHA_INICIO) & (df_6['Fecha'] <= __FECHA_FIN)]

print(df_6.head())
print(len(df_6))

      Fecha  Contratos indefinidos  Contratos temporales
4  2003-03-31           9568.0           4370.2
5  2003-06-30           9701.0           4550.9
6  2003-09-30           9816.4           4652.2
7  2003-12-31           9858.9           4687.5
8  2004-03-31           9914.9           4620.7
84

```

Juntar con el df_clean

```

In [37]: df_clean = df_clean.merge(df_6, on="Fecha", how="outer")

df_clean.head()

```

Out[37]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7

7. Poblacion

In [38]:

```
df_7 = pd.read_excel(archivo, sheet_name="Poblacion_trim")

df_7.drop(df_7.columns[[0,1,2,3,4,5,6,7,8,9]],axis=1,inplace=True)

print(df_7.head(5))
```

PERIODO	VALOR
0 2025T2	49.153.849
1 2025T1	49.077.984
2 2024T4	48.962.372
3 2024T3	48.807.137
4 2024T2	48.723.872

Pasamos a float los números

In [39]:

```
df_7 = df_7.rename(columns={"VALOR": "Poblacion")

df_7["Poblacion"] = (
    df_7["Poblacion"]
    .astype(str)
    .str.strip()
    .str.replace("\u020f", "", regex=True) # Elimina espacios finos
    .str.replace(".", "", regex=False) # Elimina puntos de miles
    .str.replace(",", ".", regex=False) # Cambia coma decimal por punto
)

df_7["Poblacion"] = pd.to_numeric(df_7["Poblacion"], errors="coerce")
```

PERIODO	Poblacion
0 2025T2	49153849
1 2025T1	49077984
2 2024T4	48962372
3 2024T3	48807137
4 2024T2	48723872

Renombramos la fecha y pasamos a tipo datetime

In [40]:

```
df_7 = df_7.rename(columns={"PERIODO": "Fecha")

df_7["Fecha"] = df_7["Fecha"].astype(str).str.strip()

df_7["Año"] = df_7["Fecha"].str[:4].astype(int)
df_7["Trimestre"] = df_7["Fecha"].str[-1:].astype(int)

df_7["Mes"] = df_7["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_7["Fecha"] = pd.to_datetime(dict(year=df_7["Año"], month=df_7["Mes"], day=1))
df_7["Fecha"] = df_7["Fecha"] + pd.offsets.QuarterEnd(0)

df_7 = df_7.drop(columns=["Año", "Trimestre", "Mes"])

df_7.head()
```

Out[40]:

	Fecha	Poblacion
0	2025-06-30	49153849
1	2025-03-31	49077984
2	2024-12-31	48962372
3	2024-09-30	48807137
4	2024-06-30	48723872

Ajustar los límites

```
In [41]: df_7 = df_7[(df_7['Fecha'] >= __FECHA_INICIO) & (df_7['Fecha'] <= __FECHA_FIN)]
print(df_7.head())
print(len(df_7))
```

	Fecha	Poblacion
6	2023-12-31	48486865
7	2023-09-30	48320520
8	2023-06-30	48205962
9	2023-03-31	48085361
10	2022-12-31	47940295

48

Juntar con el df_clean

```
In [42]: df_clean = df_clean.merge(df_7, on="Fecha", how="outer")
df_clean.head()
```

Out[42]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0

8. Cantidad de extranjeros

```
In [43]: df_8 = pd.read_excel(archivo, sheet_name="CantidadExtranjeros_trim")
df_8.drop(df_8.columns[[0,1,2,3,4,5,6,7,8,9,10,11]],axis=1,inplace=True)
df_8 = df_8.rename(columns={"VALOR": "Cantidad de extranjeros"})

df_8["Cantidad de extranjeros"] = (
    df_8["Cantidad de extranjeros"]
    .astype(str)
    .str.strip()
    .str.replace("\u202f", "", regex=True)
    .str.replace(".", "", regex=False)
    .str.replace(", ", ".", regex=False)
)

df_8["Cantidad de extranjeros"] = pd.to_numeric(df_8["Cantidad de extranjeros"], errors="coerce")
df_8 = df_8.rename(columns={"PERIODO": "Fecha"})
df_8["Fecha"] = df_8["Fecha"].astype(str).str.strip()
```

```

df_8["Año"] = df_8["Fecha"].str[:4].astype(int)
df_8["Trimestre"] = df_8["Fecha"].str[-1:].astype(int)

df_8["Mes"] = df_8["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_8["Fecha"] = pd.to_datetime(dict(year=df_8["Año"], month=df_8["Mes"], day=1))
df_8["Fecha"] = df_8["Fecha"] + pd.offsets.QuarterEnd(0)

df_8 = df_8.drop(columns=["Año", "Trimestre", "Mes"])

df_8 = df_8[(df_8['Fecha'] >= __FECHA_INICIO) & (df_8['Fecha'] <= __FECHA_FIN)]

df_clean = df_clean.merge(df_8, on="Fecha", how="outer")

df_clean.head()

```

Out[43]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

9. PIB per capita anual

In [44]:

```

df_9 = pd.read_excel(archivo, sheet_name="PIBPerCapita_Anual")

print(df_9.head(5))

```

	Fecha	PIB Per Capita	Var. anual	PIB Per Capita
0	2024	32.590 €		5,2%
1	2023	30.970 €		7,7%
2	2022	28.750 €		10,2%
3	2021	26.090 €		9,4%
4	2020	23.850 €		-10,4%

10. PIB trimestral ajustado

In [45]:

```

df_10 = pd.read_excel(archivo, sheet_name="PIB_trim_ajustados")
print(df_10.head(5))

```

	Periodo	Producto interior bruto a precios de mercado
0	2025T1	411366
1	2024T4	406745
2	2024T3	399284
3	2024T2	394493
4	2024T1	391113

In [46]:

```

df_10 = pd.read_excel(archivo, sheet_name="PIB_trim_ajustados")

# Renombrar columnas para facilitar el manejo
df_10 = df_10.rename(columns={
    "Producto interior bruto a precios de mercado": "PIB Trimestral Ajustado",
    "Periodo": "Fecha"
})

# Convertir la columna PIB a numérico
df_10["PIB Trimestral Ajustado"] = (
    df_10["PIB Trimestral Ajustado"]
    .astype(str)
    .str.strip()
    .str.replace("\u202f", "", regex=True) # Elimina espacios finos
)

```

```

        .str.replace(".", "", regex=False)      # Elimina puntos de miles
        .str.replace(",", ".", regex=False)      # Cambia coma decimal por punto
    )

df_10["PIB Trimestral Ajustado"] = pd.to_numeric(df_10["PIB Trimestral Ajustado"], errors="coerce")

# Convertir fecha de formato trimestral a datetime
df_10["Fecha"] = df_10["Fecha"].astype(str).str.strip()

df_10["Año"] = df_10["Fecha"].str[:4].astype(int)
df_10["Trimestre"] = df_10["Fecha"].str[-1:].astype(int)

df_10["Mes"] = df_10["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_10["Fecha"] = pd.to_datetime(dict(year=df_10["Año"], month=df_10["Mes"], day=1))
df_10["Fecha"] = df_10["Fecha"] + pd.offsets.QuarterEnd(0)

df_10 = df_10.drop(columns=["Año", "Trimestre", "Mes"])

# Aplicar Límites de fecha
df_10 = df_10[(df_10['Fecha'] >= __FECHA_INICIO) & (df_10['Fecha'] <= __FECHA_FIN)]

print(df_10.head(5))

```

	Fecha	PIB Trimestral Ajustado
5	2023-12-31	384835
6	2023-09-30	374053
7	2023-06-30	370217
8	2023-03-31	369227
9	2022-12-31	358462

Juntar a df_clean

```
In [47]: df_clean = df_clean.merge(df_10, on="Fecha", how="outer")

df_clean.head()
```

```
Out[47]:
```

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

11. PIB trimestral sin ajustar

```
In [48]: df_11 = pd.read_excel(archivo, sheet_name="PIB_trim_noajustados")

print(df_11.head(5))
```

	Periodo	Producto interior bruto a precios de mercado
0	2025T1	397644
1	2024T4	420435
2	2024T3	393311
3	2024T2	400313
4	2024T1	377568

```
In [49]: df_11 = df_11.rename(columns={
    "Producto interior bruto a precios de mercado": "PIB Trimestral No Ajustado",
    "Periodo": "Fecha"
})

# Convertir la columna PIB a numérico
df_11["PIB Trimestral No Ajustado"] = (
    df_11["PIB Trimestral No Ajustado"]
```

```

        .astype(str)
        .str.strip()
        .str.replace("\u202f", "", regex=True) # Elimina espacios finos
        .str.replace(".", "", regex=False) # Elimina puntos de miles
        .str.replace(", ", ".", regex=False) # Cambia coma decimal por punto
    )

df_11["PIB Trimestral No Ajustado"] = pd.to_numeric(df_11["PIB Trimestral No Ajustado"], errors="coerce")

# Convertir fecha de formato trimestral a datetime
df_11["Fecha"] = df_11["Fecha"].astype(str).str.strip()

df_11["Año"] = df_11["Fecha"].str[:4].astype(int)
df_11["Trimestre"] = df_11["Fecha"].str[-1:].astype(int)

df_11["Mes"] = df_11["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_11["Fecha"] = pd.to_datetime(dict(year=df_11["Año"], month=df_11["Mes"], day=1))
df_11["Fecha"] = df_11["Fecha"] + pd.offsets.QuarterEnd(0)

df_11 = df_11.drop(columns=["Año", "Trimestre", "Mes"])

# Aplicar límites de fecha
df_11 = df_11[(df_11['Fecha'] >= __FECHA_INICIO) & (df_11['Fecha'] <= __FECHA_FIN)]

print(df_11.head(5))

```

```

Fecha PIB Trimestral No Ajustado
5 2023-12-31 396068
6 2023-09-30 369626
7 2023-06-30 375898
8 2023-03-31 356732
9 2022-12-31 368102

```

```

In [50]: df_clean = df_clean.merge(df_11, on="Fecha", how="outer")

df_clean.head()

```

Out[50]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

12. IPI variación anual serie original

```

In [51]: df_12 = pd.read_excel(archivo, sheet_name="IPIVariacionAnualOriginal_men")

print(df_12.head(5))

```

```

Variable1          Valor1      Variable2      Valor2  \
0  Tipo de dato  Variación anual  Total Nacional  Total Nacional
1  Tipo de dato  Variación anual  Total Nacional  Total Nacional
2  Tipo de dato  Variación anual  Total Nacional  Total Nacional
3  Tipo de dato  Variación anual  Total Nacional  Total Nacional
4  Tipo de dato  Variación anual  Total Nacional  Total Nacional

Variable3          Valor3  PERIODO VALOR
0  TOTALES VARIABLES CNAE  Total industria  2025M04  -5,7
1  TOTALES VARIABLES CNAE  Total industria  2025M03  8,5
2  TOTALES VARIABLES CNAE  Total industria  2025M02  -2,3
3  TOTALES VARIABLES CNAE  Total industria  2025M01  -1,4
4  TOTALES VARIABLES CNAE  Total industria  2024M12  4,3

```

```
In [52]: # me da unos numeros raros que a ojo de buen cubero no son correctos

df_12.drop(df_12.columns[[0,1,2,3,4,5]], axis=1, inplace=True)

df_12 = df_12.rename(columns={"PERIODO": "Fecha", "VALOR": "IPI Variacion Anual Original"})

df_12["IPI Variacion Anual Original"] = (
    df_12["IPI Variacion Anual Original"]
    .astype(str)
    .str.strip()
    .str.replace("\u202f", "", regex=True)
    .str.replace(".", "", regex=False)
    .str.replace(", ", ".", regex=False)
)
df_12["IPI Variacion Anual Original"] = pd.to_numeric(df_12["IPI Variacion Anual Original"], errors="coerce")

df_12["Fecha"] = df_12["Fecha"].astype(str).str.strip()

df_12["Año"] = df_12["Fecha"].str[:4].astype(int)
df_12["Mes"] = df_12["Fecha"].str[5:].astype(int)

df_12["Fecha"] = pd.to_datetime(dict(year=df_12["Año"], month=df_12["Mes"], day=1))
df_12["Fecha"] = df_12["Fecha"] + pd.offsets.MonthEnd(0)

df_12 = df_12.drop(columns=["Año", "Mes"])

df_12 = df_12.sort_values("Fecha")

df_12 = df_12[(df_12['Fecha'] >= __FECHA_INICIO) & (df_12['Fecha'] <= __FECHA_FIN)]

df_12 = df_12.resample("QE", on="Fecha").mean().reset_index()

print(df_12.head(5))
```

```
Fecha IPI Variacion Anual Original
0 2003-03-31 3.866667
1 2003-06-30 -0.566667
2 2003-09-30 0.866667
3 2003-12-31 1.866667
4 2004-03-31 2.066667
```

```
In [53]: df_clean = df_clean.merge(df_12, on="Fecha", how="outer")
print(df_clean.head(5))
```

```
Fecha Numero Hipotecas Importe Hipotecas valor_medio_hipotecas \
0 2003-03-31 263600 2.417559e+10 91713.17
1 2003-06-30 247071 2.340044e+10 94711.40
2 2003-09-30 236977 2.397833e+10 101184.20
3 2003-12-31 241791 2.462120e+10 101828.44
4 2004-03-31 283170 3.002192e+10 106020.83

Tipo interes Medio Tasa Paro (%) Contratos indefinidos \
0 4.693333 11.99 9568.0
1 4.453333 11.28 9701.0
2 4.213333 11.30 9816.4
3 3.883333 11.37 9858.9
4 3.760000 11.50 9914.9

Contratos temporales Poblacion Cantidad de extranjeros \
0 4370.2 41827836.0 2362027.0
1 4550.9 NaN NaN
2 4652.2 42196231.0 2647431.0
3 4687.5 NaN NaN
4 4620.7 42547454.0 2900788.0

PIB Trimestral Ajustado PIB Trimestral No Ajustado \
0 195576 190941
1 199242 202908
2 202157 196602
3 205708 212242
4 209258 203881

IPI Variacion Anual Original
0 3.866667
1 -0.566667
2 0.866667
3 1.866667
4 2.066667
```

13. IPI variación anual serie corregida

```
In [54]: df_13 = pd.read_excel(archivo, sheet_name="IPIVariacionAnualCorregida_men")  
  
df_13.head(5)
```

Out[54]:

	Variable1	Valor1	Variable2	Valor2	Variable3	Valor3	Variable4	Valor4	PERIODO	VALOR
0	Tipo de dato	Variación anual	Total Nacional	Total Nacional	TOTALES VARIABLES CNAE	Total industria	Corrección de efectos	Datos ajustados de estacionalidad y calendario	2025M04	0,6
1	Tipo de dato	Variación anual	Total Nacional	Total Nacional	TOTALES VARIABLES CNAE	Total industria	Corrección de efectos	Datos ajustados de estacionalidad y calendario	2025M03	0,9
2	Tipo de dato	Variación anual	Total Nacional	Total Nacional	TOTALES VARIABLES CNAE	Total industria	Corrección de efectos	Datos ajustados de estacionalidad y calendario	2025M02	-1,8
3	Tipo de dato	Variación anual	Total Nacional	Total Nacional	TOTALES VARIABLES CNAE	Total industria	Corrección de efectos	Datos ajustados de estacionalidad y calendario	2025M01	-1,2
4	Tipo de dato	Variación anual	Total Nacional	Total Nacional	TOTALES VARIABLES CNAE	Total industria	Corrección de efectos	Datos ajustados de estacionalidad y calendario	2024M12	2

```
In [55]: df_13.drop(df_13.columns[[0,1,2,3,4,5]], axis=1, inplace=True)  
  
print("Columnas después de eliminar:", df_13.columns.tolist())  
  
df_13 = df_13.rename(columns={"PERIODO": "Fecha", "VALOR": "IPI Variacion Anual Corregida"})  
  
df_13["IPI Variacion Anual Corregida"] = (  
    df_13["IPI Variacion Anual Corregida"]  
    .astype(str)  
    .str.strip()  
    .str.replace("\u202f", "", regex=True)  
    .str.replace(".", "", regex=False)  
    .str.replace(",", ".", regex=False)  
)  
  
df_13["IPI Variacion Anual Corregida"] = pd.to_numeric(df_13["IPI Variacion Anual Corregida"], errors="coerce")  
  
# Ajustar fechas  
df_13["Fecha"] = df_13["Fecha"].astype(str).str.strip()  
df_13["Año"] = df_13["Fecha"].str[:4].astype(int)  
df_13["Mes"] = df_13["Fecha"].str[5:].astype(int)  
  
df_13["Fecha"] = pd.to_datetime(dict(year=df_13["Año"], month=df_13["Mes"], day=1))  
df_13["Fecha"] = df_13["Fecha"] + pd.offsets.MonthEnd(0)  
  
df_13 = df_13.drop(columns=["Año", "Mes"])  
  
# SELECCIONAR SOLO LAS COLUMNAS QUE NECESITAS  
df_13 = df_13[["Fecha", "IPI Variacion Anual Corregida"]]  
  
# Ordenar y filtrar fechas  
df_13 = df_13.sort_values("Fecha")  
df_13 = df_13[(df_13['Fecha'] >= __FECHA_INICIO) & (df_13['Fecha'] <= __FECHA_FIN)]  
  
# Pasar a trimestral  
df_13 = df_13.resample("QE", on="Fecha").mean().reset_index()  
  
print(df_13.head())
```

Columnas después de eliminar: ['Variable4', 'Valor4', 'PERIODO', 'VALOR']
Fecha IPI Variacion Anual Corregida
0 2003-03-31 1.433333
1 2003-06-30 1.633333
2 2003-09-30 0.866667
3 2003-12-31 1.733333
4 2004-03-31 1.866667

```
In [56]: df_clean = df_clean.merge(df_13, on="Fecha", how="outer")
```

```
df_clean.head(5)
```

Out[56]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

14. Precio medio por vivienda

```
In [57]: df_14 = pd.read_excel(archivo, sheet_name="PrecioM2Vivienda_trim")
df_14.head(5)
```

Out[57]:

	año	trimestre	€ por m2
0	Año 2003	1º	1.230,3
1	0	2º	1.309,6
2	0	3º	1.344,9
3	0	4º	1.380,3
4	Año 2004	1º	1.456,2

```
In [58]: df_14['año'] = df_14['año'].str.extract(r'(\d{4})')
df_14['año'] = df_14['año'].ffill()

df_14["Periodo"] = df_14['año'].astype(str) + df_14['trimestre'].astype(str).str.extract(r'(\d')[0]

df_14["Año"] = df_14["Periodo"].str[:4].astype(int)
df_14["Trimestre"] = df_14["Periodo"].str[-1:].astype(int)

df_14["Mes"] = df_14["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_14["Fecha"] = pd.to_datetime(dict(year=df_14["Año"], month=df_14["Mes"], day=1))
df_14["Fecha"] = df_14["Fecha"] + pd.offsets.QuarterEnd(0)

df_14 = df_14.rename(columns={df_14.columns[2]: "Precio M2 Vivienda"})

df_14["Precio M2 Vivienda"] = (
    df_14["Precio M2 Vivienda"]
    .astype(str)
    .str.replace(".", "", regex=False)
    .str.replace(",", ".", regex=False)
    .astype(float)
)

df_14 = df_14[["Fecha", "Precio M2 Vivienda"]]

df_14 = df_14[(df_14['Fecha'] >= __FECHA_INICIO) & (df_14['Fecha'] <= __FECHA_FIN)]

df_14.head(5)
```

Out[58]:

	Fecha	Precio M2 Vivienda
0	2003-03-31	1230.3
1	2003-06-30	1309.6
2	2003-09-30	1344.9
3	2003-12-31	1380.3
4	2004-03-31	1456.2

In [59]:

```
df_clean = df_clean.merge(df_14, on="Fecha", how="outer")
df_clean.head(5)
```

Out[59]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

15. Euribor a 12 meses

In [60]:

```
df_15 = pd.read_excel(archivo, sheet_name="EuriborA12Meses_mes")
df_15.columns = ["Fecha", "Euribor 12 Meses"]
df_15.head(5)
```

Out[60]:

	Fecha	Euribor 12 Meses
0	2025-02-03 00:00:00	2.504
1	2025-03-03 00:00:00	2.411
2	abr03	2.447
3	2025-05-03 00:00:00	2.252
4	2025-06-03 00:00:00	2.014

In [61]:

```
import re
from datetime import datetime
import calendar

def procesar_fecha_especial(fecha_str, posicion_global):
    """
    Procesa fechas con formato especial:
    - Cada 3 meses: formato "ene03" -> Último día del mes 2003
    - Otros meses: formato "2025-02-03" -> Último día del mes correcto
    """
    fecha_str = str(fecha_str).strip()

    meses_esp = {
        'ene': 1, 'feb': 2, 'mar': 3, 'abr': 4, 'may': 5, 'jun': 6,
        'jul': 7, 'ago': 8, 'sep': 9, 'oct': 10, 'nov': 11, 'dic': 12
    }
    if re.match(r'^[a-z]{3}\d{2}$', fecha_str.lower()):
        mes_abbr = fecha_str[:3].lower()
        año_2dig = int(fecha_str[3:])

        if mes_abbr in meses_esp:
            mes = meses_esp[mes_abbr]
            año = 2000 + año_2dig if año_2dig < 50 else 1900 + año_2dig
```

```

# Último día del mes
ultimo_dia = calendar.monthrange(año, mes)[1]
return pd.Timestamp(año, mes, ultimo_dia)

try:
    fecha_dt = pd.to_datetime(fecha_str)
    mes = fecha_dt.month
    año_base = 2003
    año_correcto = año_base + (posición_global // 12)

    # Último día del mes
    ultimo_dia = calendar.monthrange(año_correcto, mes)[1]
    return pd.Timestamp(año_correcto, mes, ultimo_dia)

except:
    return pd.NaT

df_15["Fecha_procesada"] = [
    procesar_fecha_especial(fecha, idx)
    for idx, fecha in enumerate(df_15["Fecha"])
]

print("Muestra de valores originales:", df_15["Euribor 12 Meses"].head().tolist())

def procesar_tasa_interes(valor):
    """
    Procesa tasas de interés considerando que pueden venir como:
    - "2,457" (formato europeo: 2.457%)
    - "2.457" (formato americano: 2.457%)
    - "2,457%" (con símbolo de porcentaje)
    """
    valor_str = str(valor).strip()

    valor_str = valor_str.replace("%", "")

    if "." in valor_str and "," in valor_str:
        valor_str = valor_str.replace(".", "").replace(",", ".")
    elif "," in valor_str:
        valor_str = valor_str.replace(",", ".")"

    try:
        resultado = float(valor_str)
        if -2 <= resultado <= 20:
            return resultado
        else:
            print(f"⚠️ Valor fuera de rango esperado: {resultado}")
            return resultado
    except:
        return pd.NaT

df_15["Euribor 12 Meses"] = df_15["Euribor 12 Meses"].apply(procesar_tasa_interes)

df_15["Fecha"] = df_15["Fecha_procesada"]
df_15 = df_15.drop(columns=["Fecha_procesada"])

df_15 = df_15.dropna(subset=["Fecha"])

df_15 = df_15.sort_values("Fecha")
df_15 = df_15.resample("QE", on="Fecha").mean().reset_index()

df_15 = df_15[(df_15['Fecha'] >= __FECHA_INICIO) & (df_15['Fecha'] <= __FECHA_FIN)]

```

Muestra de valores originales: [2.504, 2.411, 2.447, 2.252, 2.014]

In [62]: `df_15.head(5)`

Out[62]:

	Fecha	Euribor 12 Meses
0	2003-03-31	2.4575
1	2003-06-30	2.237667
2	2003-09-30	2.204333
3	2003-12-31	2.364667
4	2004-03-31	2.144667

```
In [63]: df_clean = df_clean.merge(df_15, on="Fecha", how="outer")
df_clean.head(5)
```

Out[63]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

16. MRO diario

```
In [64]: df_16 = pd.read_excel(archivo, sheet_name="MRO_dia")
df_16.head(15)
```

Out[64]:

	DATE	TIME PERIOD	Main refinancing operations - fixed rate tenders (fixed rate) (date of changes) - Level (FM.D.U2.EUR.4F.KR.MRR_FR.LEV)
0	1999-01-01 00:00:00	1999-01-01 00:00:00	3
1	1999-02-01 00:00:00	1999-02-01 00:00:00	3
2	1999-03-01 00:00:00	1999-03-01 00:00:00	3
3	1999-04-01 00:00:00	1999-04-01 00:00:00	3
4	1999-05-01 00:00:00	1999-05-01 00:00:00	3
5	1999-06-01 00:00:00	1999-06-01 00:00:00	3
6	1999-07-01 00:00:00	1999-07-01 00:00:00	3
7	1999-08-01 00:00:00	1999-08-01 00:00:00	3
8	1999-09-01 00:00:00	1999-09-01 00:00:00	3
9	1999-10-01 00:00:00	1999-10-01 00:00:00	3
10	1999-11-01 00:00:00	1999-11-01 00:00:00	3
11	1999-12-01 00:00:00	1999-12-01 00:00:00	3
12	13/01/1999	13/01/1999	3
13	14/01/1999	14/01/1999	3
14	15/01/1999	15/01/1999	3

```
In [65]: df_16 = df_16.rename(columns={"DATE": "Fecha", "Main refinancing operations - fixed rate tenders (fixed rate) (date of changes) - Level (FM.D.U2.EUR.4F.KR.MRR_FR.LEV)": "Cantida"})
df_16.drop(df_16.columns[[1]], axis=1, inplace=True)
```

```
In [66]: from datetime import datetime
```

```

# --- Función para procesar fechas ---
def procesar_fecha_mro(fecha_input):
    if isinstance(fecha_input, pd.Timestamp):
        return fecha_input

    if isinstance(fecha_input, (int, float)):
        try:
            return pd.to_datetime('1899-12-30') + pd.Timedelta(days=fecha_input)
        except:
            return pd.NaT

    fecha_str = str(fecha_input).strip()

    formatos = [
        '%Y-%m-%d', # 2023-01-15
        '%d/%m/%Y', # 15/01/2023
        '%m/%d/%Y', # 01/15/2023
        '%d-%m-%Y', # 15-01-2023
        '%d %b %Y', # 15 Jan 2023
        '%d-%b-%Y', # 15-Jan-2023
        '%Y%m%d', # 20230115
        '%d.%m.%Y', # 15.01.2023
        '%Y/%m/%d', # 2023/01/15
    ]

```

```

    for formato in formatos:
        try:
            return pd.to_datetime(fecha_str, format=formato)
        except:
            continue

    try:
        return pd.to_datetime(fecha_str)
    except:
        return pd.NaT

```

```

# --- Procesar fechas ---
df_16["Fecha_procesada"] = df_16["Fecha"].apply(procesar_fecha_mro)

# Mostrar posibles problemas de fecha
fechas_invalidas = df_16["Fecha_procesada"].isna().sum()
if fechas_invalidas > 0:
    problemas = df_16[df_16["Fecha_procesada"].isna()]["Fecha"].head(5)
    for i, fecha in enumerate(problemas):
        print(f" {i+1}. '{fecha}'")

```

```

df_16["Fecha"] = df_16["Fecha_procesada"]
df_16 = df_16.drop(columns=["Fecha_procesada"])

# --- Limpiar columna MRO_Rate ---
df_16["MRO_Rate"] = (
    df_16["MRO_Rate"]
    .astype(str)
    .str.strip()
    .str.replace("%", "", regex=False)
    .str.replace("\u202f", "", regex=True)
    .str.replace(",", ".", regex=False)
)
df_16["MRO_Rate"] = pd.to_numeric(df_16["MRO_Rate"], errors="coerce")

# --- Ordenar por fecha ---
df_16 = df_16.sort_values("Fecha")

# --- Reagrupar por trimestre (fin de trimestre) ---
df_16 = df_16.resample("QE", on="Fecha").mean().reset_index()

# Rellenar también al principio si faltan datos iniciales
df_16["MRO_Rate"] = df_16["MRO_Rate"].fillna(method="bfill")

# --- Rellenar MRO_Rate con el último valor anterior ---
df_16['MRO_Rate'] = df_16['MRO_Rate'].replace(0, np.nan)
df_16["MRO_Rate"] = df_16["MRO_Rate"].fillna(method="ffill")

# --- Rellenar posibles NaNs creados por resample ---
df_16["MRO_Rate"] = df_16["MRO_Rate"].fillna(method="ffill").fillna(method="bfill")

# --- Filtrar por fechas si tienes definidas las variables ---
df_16 = df_16[(df_16['Fecha'] >= __FECHA_INICIO) & (df_16['Fecha'] <= __FECHA_FIN)]

```

```
# --- Mostrar resultado final ---  
df_16.head(60)
```

```
/var/folders/cs/glwg8b695ggbr9k_ywkn40h40000gq/T/ipykernel_7142/3700456423.py:70: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.  
    df_16["MRO_Rate"] = df_16["MRO_Rate"].fillna(method="bfill")  
/var/folders/cs/glwg8b695ggbr9k_ywkn40h40000gq/T/ipykernel_7142/3700456423.py:74: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.  
    df_16["MRO_Rate"] = df_16["MRO_Rate"].fillna(method="ffill")  
/var/folders/cs/glwg8b695ggbr9k_ywkn40h40000gq/T/ipykernel_7142/3700456423.py:77: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.  
    df_16["MRO_Rate"] = df_16["MRO_Rate"].fillna(method="ffill").fillna(method="bfill")
```

Out[66]:

	Fecha	MRO_Rate
16	2003-03-31	3.500000
17	2003-06-30	3.500000
18	2003-09-30	3.500000
19	2003-12-31	3.500000
20	2004-03-31	3.500000
21	2004-06-30	3.500000
22	2004-09-30	3.500000
23	2004-12-31	3.500000
24	2005-03-31	3.500000
25	2005-06-30	3.500000
26	2005-09-30	3.500000
27	2005-12-31	3.500000
28	2006-03-31	3.500000
29	2006-06-30	3.500000
30	2006-09-30	3.500000
31	2006-12-31	3.500000
32	2007-03-31	3.500000
33	2007-06-30	3.500000
34	2007-09-30	3.500000
35	2007-12-31	3.500000
36	2008-03-31	3.500000
37	2008-06-30	3.500000
38	2008-09-30	3.500000
39	2008-12-31	3.133333
40	2009-03-31	1.680556
41	2009-06-30	1.189560
42	2009-09-30	1.133152
43	2009-12-31	1.119565
44	2010-03-31	1.000000
45	2010-06-30	1.000000
46	2010-09-30	1.000000
47	2010-12-31	1.000000
48	2011-03-31	1.100000
49	2011-06-30	1.250000
50	2011-09-30	1.399457
51	2011-12-31	1.244565
52	2012-03-31	0.958791
53	2012-06-30	0.958791
54	2012-09-30	0.807065
55	2012-12-31	0.801630
56	2013-03-31	0.683333
57	2013-06-30	0.582418
58	2013-09-30	0.527174
59	2013-12-31	0.423913

	Fecha	MRO_Rate
60	2014-03-31	0.220000
61	2014-06-30	0.200549
62	2014-09-30	0.140217
63	2014-12-31	0.093478
64	2015-03-31	0.050000
65	2015-06-30	0.050000
66	2015-09-30	0.050000
67	2015-12-31	0.050000
68	2016-03-31	0.026374
69	2016-06-30	0.004945
70	2016-09-30	0.004891
71	2016-12-31	0.004891
72	2017-03-31	0.004891
73	2017-06-30	0.004891
74	2017-09-30	0.004891
75	2017-12-31	0.004891

```
In [67]: df_clean = df_clean.merge(df_16, on="Fecha", how="outer")
df_clean.head(5)
```

Out[67]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

17. TI crédito para vivienda mensual

```
In [68]: df_17 = pd.read_excel(archivo, sheet_name="TICreditoVivienda_mes")
df_17.columns = ["Fecha", "TI_Credito"]
df_17.head(5)
```

Out[68]:

	Fecha	TI_Credito
0	2025-02-03 00:00:00	4.062
1	2025-03-03 00:00:00	3.899
2	abr03	3.786
3	2025-05-03 00:00:00	3.722
4	2025-06-03 00:00:00	3.614

```
In [69]: df_17["Fecha_procesada"] = [
    procesar_fecha_especial(fecha, idx)
    for idx, fecha in enumerate(df_17["Fecha"])
]
```

```

df_17["TI_Credito"] = df_17["TI_Credito"].apply(procesar_tasa_interes)

df_17["Fecha"] = df_17["Fecha_procesada"]
df_17 = df_17.drop(columns=["Fecha_procesada"])

df_17_original = len(df_17)
df_17 = df_17.dropna(subset=["Fecha"])

df_17 = df_17.sort_values("Fecha")
df_17 = df_17.resample("QE", on="Fecha").mean().reset_index()

print(f"Datos trimestrales: {len(df_17)}")

df_17 = df_17[(df_17['Fecha'] >= __FECHA_INICIO) & (df_17['Fecha'] <= __FECHA_FIN)]

df_17.head(5)

```

⚠ Valor fuera de rango esperado: nan
Datos trimestrales: 84

Out[69]:

	Fecha	TI_Credito
0	2003-03-31	3.9805
1	2003-06-30	3.707333
2	2003-09-30	3.363667
3	2003-12-31	3.327
4	2004-03-31	3.303333

In [70]:

```
df_clean = df_clean.merge(df_17, on="Fecha", how="outer")
df_clean.head(5)
```

Out[70]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

18. Renta disponible bruta

In [71]:

```
df_18 = pd.read_excel(archivo, sheet_name="RentaDisponibleBruta_trim")
df_18.head(5)
```

Out[71]:

	Spain	1999-Q1	Unnamed: 2	89188
0	Spain	1999-Q2	NaN	99033
1	Spain	1999-Q3	NaN	97558
2	Spain	1999-Q4	NaN	105748
3	Spain	2000-Q1	NaN	98324
4	Spain	2000-Q2	NaN	108704

In [72]:

```
df_18.drop(["Spain", "Unnamed: 2"], axis=1, inplace=True)
df_18.columns = ["Periodo", "Renta disponible bruta"]
df_18.head(5)
```

Out[72]:

	Periodo	Renta disponible bruta
0	1999-Q2	99033
1	1999-Q3	97558
2	1999-Q4	105748
3	2000-Q1	98324
4	2000-Q2	108704

```
In [73]: df_18["Renta disponible bruta"] = (
    df_18["Renta disponible bruta"]
    .astype(str)
    .str.strip()
    .str.replace("\u202f", "", regex=True) # Elimina espacios finos
    .str.replace(".", "", regex=False) # Elimina puntos de miles
    .str.replace(",", ".", regex=False) # Cambia coma decimal por punto
)

df_18["Renta disponible bruta"] = pd.to_numeric(df_18["Renta disponible bruta"], errors="coerce")

df_18["Periodo"] = df_18["Periodo"].astype(str).str.strip()

df_18["Año"] = df_18["Periodo"].str[:4].astype(int)
df_18["Trimestre"] = df_18["Periodo"].str[-1:].astype(int)
df_18["Mes"] = df_18["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

df_18["Fecha"] = pd.to_datetime(dict(year=df_18["Año"], month=df_18["Mes"], day=1))
df_18["Fecha"] = df_18["Fecha"] + pd.offsets.QuarterEnd(0)

df_18 = df_18.drop(columns=["Periodo", "Año", "Trimestre", "Mes"])

df_18 = df_18[(df_18['Fecha'] >= __FECHA_INICIO) & (df_18['Fecha'] <= __FECHA_FIN)]

df_18.head(5)
```

Out[73]:

	Renta disponible bruta	Fecha
15	120199	2003-03-31
16	139387	2003-06-30
17	125491	2003-09-30
18	138587	2003-12-31
19	127391	2004-03-31

In [74]:

```
df_clean = df_clean.merge(df_18, on="Fecha", how="outer")
df_clean.head(5)
```

Out[74]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.1
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

19. IPC mensual

```
In [75]: df_19 = pd.read_excel(archivo, sheet_name="IPC_mes")
df_19.head()
```

```
Out[75]:
```

	TIME	% var anual
0	1997-01	2,8
1	1997-02	2,5
2	1997-03	2,2
3	1997-04	1,6
4	1997-05	1,3

```
In [76]: #Renombramos las dos columnas
df_19 = df_19.rename(columns={"TIME": "Fecha", "% var anual": "IPC % Variación Anual"})
# Comprobar el tipo de dato de la columna valor
print("El formato de la columna IPC % Variación Anual es: ", df_19["IPC % Variación Anual"].dtype)

df_19.head()
```

El formato de la columna IPC % Variación Anual es: object

```
Out[76]:
```

	Fecha	IPC % Variación Anual
0	1997-01	2,8
1	1997-02	2,5
2	1997-03	2,2
3	1997-04	1,6
4	1997-05	1,3

Ajuste de la columna de variación - creo que hay que ponerle un mejor nombre

```
In [77]: #Pasar la columna % Variación Anual de object a float, viene como "2,96"
df_19["IPC % Variación Anual"] = df_19["IPC % Variación Anual"].str.replace(",",".", regex=False).astype(float)
```

Ajuste de la columna de Fecha

```
In [78]: #Asegurarse de que la columna Fechas es string y eliminar espacios sobrantes
df_19["Fecha"] = df_19["Fecha"].astype(str).str.strip()
# Viene con el formato "2023-01"
# Extrae año y mes
df_19["Año"] = df_19["Fecha"].str[:4].astype(int)
df_19["Mes"] = df_19["Fecha"].str[-2:].astype(int)
# Crea una columna de tipo datetime (día 1 de cada mes)
df_19["Fecha"] = pd.to_datetime(dict(year=df_19["Año"], month=df_19["Mes"], day=1))
# Eliminar columnas temporales
df_19 = df_19.drop(columns=["Año", "Mes"])
# Ordenar por fecha
df_19 = df_19.sort_values("Fecha")
df_19.head()
```

```
Out[78]:
```

	Fecha	IPC % Variación Anual
0	1997-01-01	2.8
1	1997-02-01	2.5
2	1997-03-01	2.2
3	1997-04-01	1.6
4	1997-05-01	1.3

Pasar a Trimestral y añadir al dataframe final

```
In [79]: # Pasamos a trimestral, en este caso, al ser una variación anual, el valor trimestral es el mismo que el mensual
df_19 = df_19.resample("QE", on="Fecha").mean().reset_index()
df_19 = df_19[(df_19['Fecha'] >= __FECHA_INICIO) & (df_19['Fecha'] <= __FECHA_FIN)]
# Hacer merge con el DataFrame existente
df_clean = df_clean.merge(df_19, on="Fecha", how="outer")
df_clean.head()
```

Out[79]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

20. Número de compraventas

```
In [80]: df_20 = pd.read_excel(archivo, sheet_name="NumCompraventas_trim")
df_20.head()
```

Out[80]:

	año	trimestre	valores absolutos
0	Año 2004	1º	190.442
1	0	2º	223.895
2	0	3º	201.089
3	0	4º	232.964
4	Año 2005	1º	196.438

Ajuste de la tabla a forma correcta

```
In [81]: # Necesito ajustar las columnas para tener una columna de fecha y una de valor, de momento el formato es el siguiente
# año trimestre valores absolutos
# 0 Año 2004 1º 190.442
# 1 0 2º 223.895
# 2 0 3º 201.089
# 3 0 4º 232.964
# 4 Año 2005 1º 196.438

# En primer lugar, añadir una columna año con el año correspondiente de cada dato, ten en cuenta que el formato es el siguiente
df_20['año'] = df_20['año'].str.extract(r'(\d{4})') # Extrae el año si hay texto tipo "Año 2004"
df_20['año'] = df_20['año'].fillna(method='ffill')

df_20.head()
```

```
/var/folders/cs/glwgb695ggbr9k_ywkn40h40000gq/T/ipykernel_7142/3888055254.py:11: FutureWarning: Series.fillna will use 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
df_20['año'] = df_20['año'].fillna(method='ffill')
```

Out[81]:

	año	trimestre	valores absolutos
0	2004	1º	190.442
1	2004	2º	223.895
2	2004	3º	201.089
3	2004	4º	232.964
4	2005	1º	196.438

Ajuste de la columna de Fecha

```
In [82]: # 2. Crear columna "Periodo" tipo "YYYYT" (ej: 20041, 20042, ...)
df_20["Periodo"] = df_20['año'].astype(str) + df_20['trimestre'].astype(str).str.extract(r'(\d)')[0]

# 3. Crear columnas Año y Trimestre
df_20["Año"] = df_20["Periodo"].str[:4].astype(int)
df_20["Trimestre"] = df_20["Periodo"].str[-1:].astype(int)

# 4. Mapear trimestre a mes final de trimestre
df_20["Mes"] = df_20["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

# 5. Crear columna Fecha tipo datetime al final de cada trimestre
df_20["Fecha"] = pd.to_datetime(dict(year=df_20["Año"], month=df_20["Mes"], day=1))
df_20["Fecha"] = df_20["Fecha"] + pd.offsets.QuarterEnd(0)

# 6. Límpiar y dejar solo las columnas necesarias
df_20 = df_20.rename(columns={"valores absolutos": "Número de compraventas"})
df_20["Número de compraventas"] = df_20["Número de compraventas"].astype(str).str.replace('.', '', regex=False).str.replace(',', '.', regex=False)
df_20 = df_20[["Fecha", "Número de compraventas"]]

df_20.head()
```

Out[82]:

	Fecha	Número de compraventas
0	2004-03-31	190442
1	2004-06-30	223895
2	2004-09-30	201089
3	2004-12-31	232964
4	2005-03-31	196438

Añadir al dataframe final

```
In [83]: df_20 = df_20[(df_20['Fecha'] >= __FECHA_INICIO) & (df_20['Fecha'] <= __FECHA_FIN)]
# Hacer merge con el DataFrame existente
df_clean = df_clean.merge(df_20, on="Fecha", how="outer")
df_clean.head()
```

Out[83]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows x 21 columns

21. Valor de compraventas

```
In [84]: df_21 = pd.read_excel(archivo, sheet_name="ValorCompraventa_trim")
df_21.head()
```

Out[84]:

	año	trimestre	miles de euros
0	Año 2004	1º	22.184.921,1
1	0	2º	28.377.885,1
2	0	3º	25.815.818,4
3	0	4º	30.504.462,2
4	Año 2005	1º	27.268.283,4

Ajuste de la tabla a forma correcta

In [85]:

```
# En primer lugar, añadir una columna año con el año correspondiente de cada dato, ten en cuenta que el formato es incorrecto
df_21['año'] = df_21['año'].str.extract(r'(\d{4})') # Extrae el año si hay texto tipo "Año 2004"
df_21['año'] = df_21['año'].fillna(method='ffill')

df_21.head()
```

/var/folders/cs/glwg8b695ggbr9k_ywkn40h40000gq/T/ipykernel_7142/3985589348.py:3: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.fillna() or obj.bfill() instead.
df_21['año'] = df_21['año'].fillna(method='ffill')

Out[85]:

	año	trimestre	miles de euros
0	2004	1º	22.184.921,1
1	2004	2º	28.377.885,1
2	2004	3º	25.815.818,4
3	2004	4º	30.504.462,2
4	2005	1º	27.268.283,4

In [86]:

```
# 2. Crear columna "Periodo" tipo "YYYYT" (ej: 20041, 20042, ...)
df_21["Periodo"] = df_21['año'].astype(str) + df_21['trimestre'].astype(str).str.extract(r'(\d)')[0]

# 3. Crear columnas Año y Trimestre
df_21["Año"] = df_21["Periodo"].str[:4].astype(int)
df_21["Trimestre"] = df_21["Periodo"].str[-1:].astype(int)

# 4. Mapear trimestre a mes final de trimestre
df_21["Mes"] = df_21["Trimestre"].map({1: 3, 2: 6, 3: 9, 4: 12})

# 5. Crear columna Fecha tipo datetime al final de cada trimestre
df_21["Fecha"] = pd.to_datetime(dict(year=df_21["Año"], month=df_21["Mes"], day=1))
df_21["Fecha"] = df_21["Fecha"] + pd.offsets.QuarterEnd(0)

# 6. Limpiar y dejar solo las columnas necesarias
# Ten en cuenta que el valor viene en miles de euros, por lo que hay que multiplicar por 1000 y viene con el formato incorrecto
df_21 = df_21.rename(columns={"miles de euros": "Valor de compraventa"})
df_21["Valor de compraventa"] = (
    df_21["Valor de compraventa"]
    .astype(str)
    .str.replace(".", "", regex=False) # Elimina puntos de miles
    .str.replace(",", ".", regex=False) # Cambia coma decimal por punto
    .astype(float) * 1000 # Multiplica por 1000 para convertir a euros
)
df_21 = df_21[["Fecha", "Valor de compraventa"]]
df_21 = df_21[(df_21['Fecha'] >= __FECHA_INICIO) & (df_21['Fecha'] <= __FECHA_FIN)]

df_21.head()
```

Out[86]:

	Fecha	Valor de compraventa
0	2004-03-31	2.218492e+10
1	2004-06-30	2.837789e+10
2	2004-09-30	2.581582e+10
3	2004-12-31	3.050446e+10
4	2005-03-31	2.726828e+10

```
In [87]: # Hacer merge con el DataFrame existente
df_clean = df_clean.merge(df_21, on="Fecha", how="outer")
df_clean.head()
```

Out[87]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 22 columns

◀ ▶

22. Interés Fijo Variable / Anual

```
In [88]: df_22 = pd.read_excel(archivo, sheet_name="interes_fijo_variable_anual")
df_22.head()
```

Out[88]:

	Fijo	Variable
0	2024	3,42
1	2023	2,86
2	2022	2,04
3	2021	1,91
4	2020	2,31

```
In [89]: df_22.dtypes
```

```
Out[89]:
```

	int64
Fijo	object
Variable	object
dtype:	object

Ajuste de las columnas numéricas

```
In [90]: # Transformar la columna "Fijo" a float
df_22["Interes_Fijo_Hipotecas"] = df_22["Fijo"].astype(str).str.replace(",",".", regex=False).astype(float)
# Transformar la columna "Variable" a float
df_22["Interes_Variable_Hipotecas"] = df_22["Variable"].astype(str).str.replace(",",".", regex=False).astype(float)
df_22.dtypes
```

```
Out[90]:
```

	int64
Fijo	object
Variable	object
Interes_Fijo_Hipotecas	float64
Interes_Variable_Hipotecas	float64
dtype:	object

```
In [91]: df_22 = df_22.drop(["Fijo", "Variable"], axis=1)
df_22.head()
```

Out[91]:

	Interes_Fijo_Hipotecas	Interes_Variable_Hipotecas
0	2024	3.42
1	2023	3.30
2	2022	2.04
3	2021	1.91
4	2020	2.31

Ajuste de la columna Fecha

In [92]:

```
# El header de la columna "Fecha" está vacío, lo renombramos por el índice de la columna
df_22.columns = ["Fecha", "Interes_Fijo_Hipotecas", "Interes_Variable_Hipotecas"]
# Convertir la columna Fecha a datetime, es anual, por lo que solo nos indica el año
df_22["Fecha"] = pd.to_datetime(df_22["Fecha"].astype(str) + "-01-01")
df_22.head()
```

Out[92]:

	Fecha	Interes_Fijo_Hipotecas	Interes_Variable_Hipotecas
0	2024-01-01	3.42	3.07
1	2023-01-01	3.30	2.86
2	2022-01-01	2.04	1.90
3	2021-01-01	1.91	1.94
4	2020-01-01	2.31	2.12

In [93]:

```
# Convertir la columna Fecha a trimestral, como es anual, todos los trimestres tendrán el valor del año
df_22 = df_22.set_index("Fecha").resample("QE").ffill().reset_index()
# Filtrar por las fechas de inicio y fin
df_22 = df_22[(df_22['Fecha'] >= __FECHA_INICIO) & (df_22['Fecha'] <= __FECHA_FIN)]
df_22.head()
```

Out[93]:

	Fecha	Interes_Fijo_Hipotecas	Interes_Variable_Hipotecas
0	2003-03-31	5.15	4.29
1	2003-06-30	5.15	4.29
2	2003-09-30	5.15	4.29
3	2003-12-31	5.15	4.29
4	2004-03-31	4.29	3.59

Añadir al dataframe final

In [94]:

```
# Hacer merge con el DataFrame existente
df_clean = df_clean.merge(df_22, on="Fecha", how="outer")
df_clean.head()
```

Out[94]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 11 columns

24. Índice de confianza del consumidor / Mes

In [95]:

```
# Leemos la hoja de afiliación a la seguridad social
df_24 = pd.read_excel(archivo, sheet_name="indice_conf_consumidor")
df_24.head()
```

Out[95]:

	Año	Periodo	Confianza del consumidor
0	2004	Septiembre	96,69
1	2004	Octubre	90,64
2	2004	Noviembre	90,05
3	2004	Diciembre	92,11
4	2005	Enero	98,03

Conversión de tipos

In [96]:

```
# Verificamos el tipo de dato de todas las columnas
df_24.dtypes
```

Out[96]:

Año	int64
Periodo	object
Confianza del consumidor	object
dtype:	object

In [97]:

```
# Convertimos Confianza del consumidor a float, ya que es un índice
df_24["Confianza del consumidor"] = df_24["Confianza del consumidor"].str.replace(",",".", regex=False).astype(float)
# Renombramos la columna
df_24 = df_24.rename(columns={"Confianza del consumidor": "Confianza_consumidor"})
df_24.head()
```

Out[97]:

	Año	Periodo	Confianza_consumidor
0	2004	Septiembre	96.69
1	2004	Octubre	90.64
2	2004	Noviembre	90.05
3	2004	Diciembre	92.11
4	2005	Enero	98.03

Ajuste de columna Fecha

In [98]:

```
# Ahora tenemos que crear la columna Fecha, tenemos una columna Año y una columna Periodo que nos muestra "Septiembre", "Octubre", "Noviembre", "Diciembre", "Enero"
# Primero tenemos que convertir el mes a un formato numérico, por lo que creamos un diccionario de meses
meses = {
    "Enero": "01", "Febrero": "02", "Marzo": "03", "Abril": "04",
```

```

        "Mayo": "05", "Junio": "06", "Julio": "07", "Agosto": "08",
        "Septiembre": "09", "Octubre": "10", "Noviembre": "11", "Diciembre": "12"
    }

# Reemplazamos los nombres de los meses por su número correspondiente
df_24["Periodo"] = df_24["Periodo"].map(meses)
df_24.head()

```

Out[98]:

	Año	Periodo	Confianza_consumidor
0	2004	09	96.69
1	2004	10	90.64
2	2004	11	90.05
3	2004	12	92.11
4	2005	01	98.03

In [99]:

```

# Tenemos la columna Año y la columna Periodo con el mes en formato numérico, ahora creamos la columna Fecha
df_24["Fecha"] = pd.to_datetime(df_24["Año"].astype(str) + "-" + df_24["Periodo"], format="%Y-%m")
# Eliminamos las columnas Año y Periodo
df_24 = df_24.drop(columns=["Año", "Periodo"])
df_24.head()

```

Out[99]:

	Confianza_consumidor	Fecha
0	96.69	2004-09-01
1	90.64	2004-10-01
2	90.05	2004-11-01
3	92.11	2004-12-01
4	98.03	2005-01-01

In [100]:

```

# Pasamos a trimestral, en este caso, al ser un índice, el valor trimestral vamos a calcularlo como la media de los tres meses
df_24 = df_24.resample("Q", on="Fecha").mean().reset_index()
# Filtramos por las fechas de inicio y fin
df_24 = df_24[(df_24['Fecha'] >= __FECHA_INICIO) & (df_24['Fecha'] <= __FECHA_FIN)]
df_24.head(50)

```

Out[100]:

	Fecha	Confianza_consumidor
0	2004-09-30	96.690000
1	2004-12-31	90.933333
2	2005-03-31	94.920000
3	2005-06-30	91.210000
4	2005-09-30	90.400000
5	2005-12-31	85.640000
6	2006-03-31	88.826667
7	2006-06-30	84.916667
8	2006-09-30	86.836667
9	2006-12-31	88.790000
10	2007-03-31	89.856667
11	2007-06-30	93.580000
12	2007-09-30	86.383333
13	2007-12-31	74.876667
14	2008-03-31	73.573333
15	2008-06-30	57.320000
16	2008-09-30	49.086667
17	2008-12-31	49.250000
18	2009-03-31	50.803333
19	2009-06-30	64.016667
20	2009-09-30	75.316667
21	2009-12-31	73.076667
22	2010-03-31	74.166667
23	2010-06-30	69.726667
24	2010-09-30	73.756667
25	2010-12-31	65.835000
26	2011-03-31	70.800000
27	2011-06-30	74.133333
28	2011-09-30	70.966667
29	2011-12-31	67.300000
30	2012-03-31	65.985000
31	2012-06-30	50.450000
32	2012-09-30	40.800000
33	2012-12-31	44.550000
34	2013-03-31	52.833333
35	2013-06-30	56.433333
36	2013-09-30	67.233333
37	2013-12-31	68.800000
38	2014-03-31	75.166667
39	2014-06-30	87.100000
40	2014-09-30	88.633333
41	2014-12-31	87.000000
42	2015-03-31	100.000000
43	2015-06-30	102.100000

	Fecha	Confianza_consumidor
44	2015-09-30	105.866667
45	2015-12-31	103.926667
46	2016-03-31	95.623333
47	2016-06-30	93.490000
48	2016-09-30	94.353333
49	2016-12-31	96.916667

Pasamos al dataframe final

```
In [101... # Hacer merge con el DataFrame existente
df_clean = df_clean.merge(df_24, on="Fecha", how="outer")
df_clean.head()
```

```
Out[101...  

      Fecha  Numero_Hipotecas  Importe_Hipotecas  valor_medio_hipotecas  Tipo_interes_Medio  Tasa_Paro_(%)  Contratos_indefinidos  Contratos_temporales  Poblacion  Cantidad_de_extranjero  

0  2003-03-31  263600  2.417559e+10  91713.17  4.693333  11.99  9568.0  4370.2  41827836.0  2362027.0  

1  2003-06-30  247071  2.340044e+10  94711.40  4.453333  11.28  9701.0  4550.9  NaN  NaN  

2  2003-09-30  236977  2.397833e+10  101184.20  4.213333  11.30  9816.4  4652.2  42196231.0  2647431.0  

3  2003-12-31  241791  2.462120e+10  101828.44  3.883333  11.37  9858.9  4687.5  NaN  NaN  

4  2004-03-31  283170  3.002192e+10  106020.83  3.760000  11.50  9914.9  4620.7  42547454.0  2900788.0
```

5 rows × 25 columns

25. Afiliación a la seguridad social / Mes

```
In [102... # Leemos la hoja de afiliación a la seguridad social
df_25 = pd.read_excel(archivo, sheet_name="afiliacion_ss")
df_25.head()
```

```
Out[102...  

      Periodo  TOTAL_SISTEMA  CUIDADORES_(10)  

0  NaN  

1  Enero 1985  10.457.345  NaN  

2  Febrero 1985  10.433.718  NaN  

3  Marzo 1985  10.422.878  NaN  

4  Abril 1985  10.452.983  NaN
```

Eliminación primera fila

```
In [103... df_25 = df_25.drop(index=0)
df_25.head()
```

Out[103...]

	Periodo	TOTAL SISTEMA	CUIDADORES (10)
1	Enero 1985	10.457.345	NaN
2	Febrero 1985	10.433.718	NaN
3	Marzo 1985	10.422.878	NaN
4	Abril 1985	10.452.983	NaN
5	Mayo 1985	10.498.490	NaN

Conversión de tipos

In [104...]

```
# Verificamos el tipo de dato de todas las columnas
df_25.dtypes
```

Out[104...]

	Periodo	TOTAL SISTEMA	CUIDADORES (10)
		object	object
		object	object
		object	object

In [105...]

```
# Convertimos las columnas TOTAL SISTEMA y CUIDADORES (10) a float, pero también tenemos que eliminar los puntos que separan las decimales
df_25["TOTAL SISTEMA"] = df_25["TOTAL SISTEMA"].astype(str).str.replace(".", "", regex=False).astype(float)
df_25["CUIDADORES (10)"] = df_25["CUIDADORES (10)"].astype(str).str.replace(".", "", regex=False).astype(float)
# Comprobamos de nuevo el tipo de datos
df_25.dtypes
```

Out[105...]

	Periodo	TOTAL SISTEMA	CUIDADORES (10)
		object	float64
		float64	float64
		object	object

In [106...]

```
# Convertimos los NaN de CUIDADORES (10) a ceros
df_25["CUIDADORES (10)"] = df_25["CUIDADORES (10)"].fillna(0)
df_25.head()
# Sumamos las columnas TOTAL SISTEMA Y CUIDADORES (10) para obtener el total de afiliaciones a la seguridad social
df_25["Afiliaciones_SS"] = df_25["TOTAL SISTEMA"] + df_25["CUIDADORES (10)"]
df_25.head()
# Eliminamos TOTAL SISTEMA y CUIDADORES (10)
df_25 = df_25.drop(columns=["TOTAL SISTEMA", "CUIDADORES (10)"])
df_25.head()
```

Out[106...]

	Periodo	Afiliaciones_SS
1	Enero 1985	10457345.0
2	Febrero 1985	10433718.0
3	Marzo 1985	10422878.0
4	Abril 1985	10452983.0
5	Mayo 1985	10498490.0

In [107...]

```
# Diccionario de traducción de meses del español al inglés
meses_es_en = {
    "Enero": "January",
    "Febrero": "February",
    "Marzo": "March",
    "Abril": "April",
    "Mayo": "May",
    "Junio": "June",
    "Julio": "July",
    "Agosto": "August",
    "Septiembre": "September",
    "Octubre": "October",
    "Noviembre": "November",
    "Diciembre": "December"
}

# Eliminar espacios en blanco al inicio/final de cada string (ha saltado error en una fila por no hacer esto esto)
df_25["Periodo"] = df_25["Periodo"].astype(str).str.strip()

# Reemplazar nombres de meses por su equivalente en inglés
df_25["Fecha"] = df_25["Periodo"].replace(meses_es_en, regex=True)

# Convertir a datetime y quedarnos con inicio de mes
```

```

df_25["Fecha"] = pd.to_datetime(df_25["Fecha"], format="%B %Y").dt.to_period("M").dt.to_timestamp()

# Eliminar columna original y reordenar
df_25 = df_25.drop(columns=["Periodo"])
df_25 = df_25[['Fecha', "Afiliaciones_SS"]]

df_25.head()

```

Out[107...]

	Fecha	Afiliaciones_SS
1	1985-01-01	10457345.0
2	1985-02-01	10433718.0
3	1985-03-01	10422878.0
4	1985-04-01	10452983.0
5	1985-05-01	10498490.0

Cambiar periodicidad y límite temporal

In [108...]

```

# Nos aseguramos que las fechas estén ordenadas
df_25 = df_25.sort_values("Fecha")
# Pasar a trimestral (suma en este caso)
df_25 = df_25.resample("QE", on="Fecha").sum().reset_index()
# Ajustamos al límite temporal del análisis
df_25 = df_25[(df_25['Fecha'] >= __FECHA_INICIO) & (df_25['Fecha'] <= __FECHA_FIN)]
df_25.head()

```

Out[108...]

	Fecha	Afiliaciones_SS
72	2003-03-31	49006568.0
73	2003-06-30	50002323.0
74	2003-09-30	50159070.0
75	2003-12-31	50194767.0
76	2004-03-31	50448144.0

Añadimos columna al Dataframe final

In [109...]

```

df_clean = df_clean.merge(df_25, on="Fecha", how="outer")
df_clean.head()

```

Out[109...]

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 26 columns

◀ ▶

26. Plazo medio devolución de hipotecas anual / mes

In [110...]

```

# Leemos la hoja de plazo medio de devolución de hipotecas
df_26 = pd.read_excel(archivo, sheet_name="plazo_hipotecas")
df_26.head(50)

```

Out[110...]

	Fecha	Años
0	2003	23
1	2004	24
2	2005	25
3	2006	27
4	2007	28
5	2008	27
6	2009M01	25
7	2009M02	25
8	2009M03	24
9	2009M04	24
10	2009M05	24
11	2009M06	24
12	2009M07	24
13	2009M08	23
14	2009M09	24
15	2009M10	24
16	2009M11	24
17	2009M12	24
18	2010M01	24
19	2010M02	24
20	2010M03	24
21	2010M04	24
22	2010M05	24
23	2010M06	25
24	2010M07	24
25	2010M08	25
26	2010M09	25
27	2010M10	23
28	2010M11	25
29	2010M12	24
30	2011M01	25
31	2011M02	26
32	2011M03	24
33	2011M04	24
34	2011M05	23
35	2011M06	24
36	2011M07	24
37	2011M08	23
38	2011M09	23
39	2011M10	23
40	2011M11	23
41	2011M12	24
42	2012M01	23
43	2012M02	24

	Fecha	Años
44	2012M03	23
45	2012M04	26
46	2012M05	23
47	2012M06	23
48	2012M07	23
49	2012M08	24

Conversión de tipos

```
In [111...]: # Comprobamos el tipo de todas las columnas
df_26.dtypes
# Convertimos la columna Años a float y la renombramos a Duracion_media_hipoteca
df_26[ "Años" ] = df_26[ "Años" ].astype( float )
df_26 = df_26.rename( columns={ "Años": "Duracion_media_hipoteca" } )
df_26.head( )
```

```
Out[111...]:
```

	Fecha	Duracion_media_hipoteca
0	2003	23.0
1	2004	24.0
2	2005	25.0
3	2006	27.0
4	2007	28.0

```
In [112...]: # Ahora convertimos la columna Fecha. Los datos hasta 2009 son anuales y los vamos a convertir al primer día del año
df_26[ "Fecha_temp" ] = df_26[ "Fecha" ].astype( str ).apply( lambda x: x + "-01-01" if len( x ) == 4 and x.isdigit() else x )
df_26[ "Fecha_convertida" ] = pd.to_datetime( df_26[ "Fecha_temp" ], errors='coerce', format="%Y-%m-%d" )
mask_nan = df_26[ "Fecha_convertida" ].isna( )
df_26.loc[ mask_nan, "Fecha_convertida" ] = pd.to_datetime(
    df_26.loc[ mask_nan, "Fecha" ].str.slice( 0, 4 ) + "-" + df_26.loc[ mask_nan, "Fecha" ].str.slice( 5, 7 ) + "-01",
    format="%Y-%m-%d",
    errors='coerce'
)
df_26[ "Fecha_formateada" ] = df_26[ "Fecha_convertida" ].dt.strftime( "%d-%m-%Y" )
# Eliminamos las columnas Fecha_temp y Fecha_formateada y Fecha. Renombramos Fecha_convertida a Fecha
df_26.drop( columns=[ "Fecha", "Fecha_temp", "Fecha_formateada" ], inplace=True )
df_26.rename( columns={ "Fecha_convertida": "Fecha" }, inplace=True )
# Cambiamos orden de las columnas
df_26 = df_26[ [ "Fecha", "Duracion_media_hipoteca" ] ]
df_26.head( 10 )
```

```
Out[112...]:
```

	Fecha	Duracion_media_hipoteca
0	2003-01-01	23.0
1	2004-01-01	24.0
2	2005-01-01	25.0
3	2006-01-01	27.0
4	2007-01-01	28.0
5	2008-01-01	27.0
6	2009-01-01	25.0
7	2009-02-01	25.0
8	2009-03-01	24.0
9	2009-04-01	24.0

Cambiar periodicidad y límite temporal

```
In [113...]: # Nos aseguramos que las fechas estén ordenadas
df_26[ "Fecha" ] = pd.to_datetime( df_26[ "Fecha" ] )
df_26 = df_26.sort_values( "Fecha" )
```

```
# Pasar a trimestral (prmedio en este caso)
df_26 = df_26.resample("QE", on="Fecha").mean(numeric_only=True).round(2).reset_index()
# Ajustamos al Límite temporal del análisis
df_26 = df_26[(df_26['Fecha'] >= __FECHA_INICIO) & (df_26['Fecha'] <= __FECHA_FIN)]
df_26.head(30)
```

Out[113...]

	Fecha	Duracion_media_hipoteca
0	2003-03-31	23.00
1	2003-06-30	NaN
2	2003-09-30	NaN
3	2003-12-31	NaN
4	2004-03-31	24.00
5	2004-06-30	NaN
6	2004-09-30	NaN
7	2004-12-31	NaN
8	2005-03-31	25.00
9	2005-06-30	NaN
10	2005-09-30	NaN
11	2005-12-31	NaN
12	2006-03-31	27.00
13	2006-06-30	NaN
14	2006-09-30	NaN
15	2006-12-31	NaN
16	2007-03-31	28.00
17	2007-06-30	NaN
18	2007-09-30	NaN
19	2007-12-31	NaN
20	2008-03-31	27.00
21	2008-06-30	NaN
22	2008-09-30	NaN
23	2008-12-31	NaN
24	2009-03-31	24.67
25	2009-06-30	24.00
26	2009-09-30	23.67
27	2009-12-31	24.00
28	2010-03-31	24.00
29	2010-06-30	24.33

Añadimos columna al Dataframe final

In [114...]

```
df_clean = df_clean.merge(df_26, on="Fecha", how="outer")
df_clean.head()
```

Out[114...]

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 27 columns

---- Fin recolección variables. Pequeños reajustes en dataset final ----

Comprobación de nulos por variable

In [115...]

```
print(analizar_nan(df_clean))
```

	Columna	NaN_Count	NaN_Percentage	Datos_Válidos	\
8	Poblacion	36	42.86	48	
9	Cantidad de extranjeros	36	42.86	48	
26	Duracion_media_hipoteca	18	21.43	66	
24	Confianza_consumidor	6	7.14	78	
21	Valor de compraventa	4	4.76	80	
20	Número de compraventas	4	4.76	80	
15	Euribor 12 Meses	0	0.00	84	
25	Afiliaciones_SS	0	0.00	84	
23	Interes_Variable_Hipotecas	0	0.00	84	
22	Interes_Fijo_Hipotecas	0	0.00	84	
19	IPC % Variación Anual	0	0.00	84	
18	Renta disponible bruta	0	0.00	84	
17	TI_Credito	0	0.00	84	
16	MRO_Rate	0	0.00	84	
0	Fecha	0	0.00	84	
14	Precio M2 Vivienda	0	0.00	84	
1	Numero Hipotecas	0	0.00	84	
12	IPI Variacion Anual Original	0	0.00	84	
11	PIB Trimestral No Ajustado	0	0.00	84	
10	PIB Trimestral Ajustado	0	0.00	84	
7	Contratos temporales	0	0.00	84	
6	Contratos indefinidos	0	0.00	84	
5	Tasa Paro (%)	0	0.00	84	
4	Tipo interes Medio	0	0.00	84	
3	valor_medio_hipotecas	0	0.00	84	
2	Importe Hipotecas	0	0.00	84	
13	IPI Variacion Anual Corregida	0	0.00	84	
	Total_Filas				
8		84			
9		84			
26		84			
24		84			
21		84			
20		84			
15		84			
25		84			
23		84			
22		84			
19		84			
18		84			
17		84			
16		84			
0		84			
14		84			
1		84			
12		84			
11		84			
10		84			
7		84			
6		84			
5		84			
4		84			
3		84			
2		84			
13		84			

Prepocesamiento Nulos

1. Número de compraventas y valor de compraventas

Vamos a rellenar los NaN de ambas variables calculando las tasas de crecimiento trimestrales desde 2004 (año en el que empieza a haber datos) hasta 2006 (anterior a la crisis). Escogemos este período para que las tasas de crecimiento no se desvirtúen por el período posterior a la crisis.

```
In [116]: df_20 = df_20.sort_values("Fecha").set_index("Fecha").asfreq("QE")

# Calculamos la tasa de crecimiento trimestral (pct_change)
df_20_crecimiento = df_20["Número de compraventas"].pct_change()

# Calculamos el crecimiento medio solo entre 2004 y 2007
crecimiento_medio = df_20_crecimiento["2004":"2007"].mean()

primer_valor_2004 = df_20["Número de compraventas"]["2004-03-31"]

# Calculamos valores 2003 aplicando crecimiento medio hacia atrás
```

```

valores_2003 = [primer_valor_2004 / ((1 + crecimiento_medio) ** i) for i in range(1, 5)]
fechas_2003 = pd.date_range(end="2003-12-31", periods=4, freq="QE")

df_2003 = pd.DataFrame({
    "Número de compraventas": valores_2003[::-1]
}, index=fechas_2003)

df_20 = pd.concat([df_2003, df_20]).sort_index().reset_index().rename(columns={"index": "Fecha"})

```

In [117... df_20.head()

Out[117... **Fecha** **Número de compraventas**

0	2003-03-31	183908.527902
1	2003-06-30	185520.576206
2	2003-09-30	187146.754903
3	2003-12-31	188787.187852
4	2004-03-31	190442.000000

In [118... df_21 = df_21.sort_values("Fecha").set_index("Fecha").asfreq("QE")

```

# Calculamos la tasa de crecimiento trimestral (pct_change)
df_21_crecimiento = df_21["Valor de compraventa"].pct_change()

# Calculamos el crecimiento medio solo entre 2004 y 2007
crecimiento_medio = df_21_crecimiento["2004":"2007"].mean()

primer_valor_2004 = df_21["Valor de compraventa"]["2004-03-31"]

# Calculamos valores 2003 aplicando crecimiento medio hacia atrás
valores_2003 = [primer_valor_2004 / ((1 + crecimiento_medio) ** i) for i in range(1, 5)]
fechas_2003 = pd.date_range(end="2003-12-31", periods=4, freq="QE")

df_2003 = pd.DataFrame({
    "Valor de compraventa": valores_2003[::-1]
}, index=fechas_2003)

df_21 = pd.concat([df_2003, df_21]).sort_index().reset_index().rename(columns={"index": "Fecha"})

```

In [119... df_21.head()

Out[119... **Fecha** **Valor de compraventa**

0	2003-03-31	1.910958e+10
1	2003-06-30	1.983594e+10
2	2003-09-30	2.058991e+10
3	2003-12-31	2.137254e+10
4	2004-03-31	2.218492e+10

In [120... df_clean = df_clean.merge(df_20, on="Fecha", how="outer")
df_clean = df_clean.merge(df_21, on="Fecha", how="outer")
df_clean.head()

Out[120...]

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 29 columns

In [121...]

```
df_clean = df_clean.drop(columns=["Número de compraventas_x", "Valor de compraventa_x"])
df_clean.head()
```

Out[121...]

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 27 columns

In [122...]

```
df_clean = df_clean.rename(columns={"Número de compraventas_y": "Numero_compraventas"})
df_clean = df_clean.rename(columns={"Valor de compraventa_y": "Valor_compraventa"})
df_clean.head()
```

Out[122...]

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows × 27 columns

In [123...]

```
import matplotlib.pyplot as plt

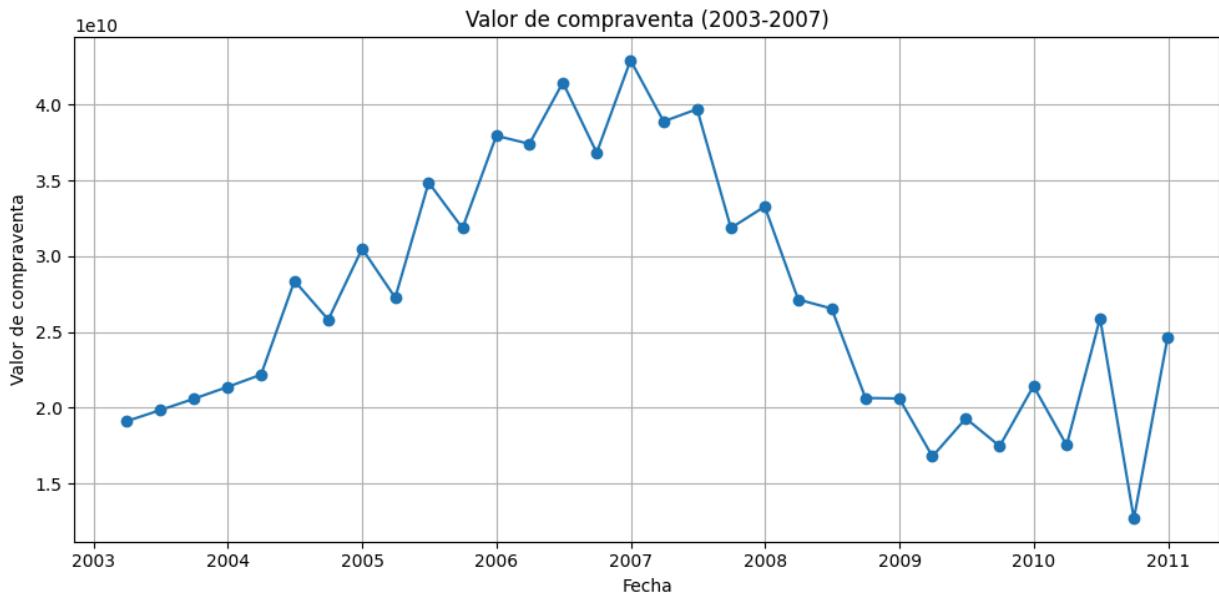
# Filtrar los datos entre 2003 y 2007
mask = (df_clean['Fecha'] >= '2003-01-01') & (df_clean['Fecha'] <= '2010-12-31')
df_plot = df_clean.loc[mask]

plt.figure(figsize=(10,5))
```

```

plt.plot(df_plot['Fecha'], df_plot['Valor_compraventa'], marker='o', linestyle='-' )
plt.title('Valor de compraventa (2003-2007)')
plt.xlabel('Fecha')
plt.ylabel('Valor de compraventa')
plt.grid(True)
plt.tight_layout()
plt.show()

```



2. Duración Media Hipoteca

En este caso, tenemos variables trimestrales desde 2009, y anuales de 2003 a 2009. Esta variable posee valores muy similares a lo largo del año, por lo que podemos sustituir estos valores anuales en los trimestres restantes de cada año sin introducir sesgos significativos.

Al tratarse de una métrica estructural como la duración media de una hipoteca, que tiende a cambiar de forma lenta y progresiva, no tiene sentido aplicar interpolación entre trimestres cuando sólo disponemos de un dato por año. Además, así evitamos fluctuaciones no reales

```

In [124... df_clean['Duracion_media_hipoteca'] = df_clean.apply(
    lambda row: df_clean[
        (df_clean['Fecha'].dt.year == row['Fecha'].year) &
        (df_clean['Fecha'].dt.quarter == 1)
    ]['Duracion_media_hipoteca'].values[0]
    if row['Fecha'].year < 2009 and pd.isna(row['Duracion_media_hipoteca']) else row['Duracion_media_hipoteca'],
    axis=1
)
df_clean.head()

```

Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjero
2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836.0	2362027.0
2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	NaN	NaN
2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231.0	2647431.0
2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	NaN	NaN
2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454.0	2900788.0

5 rows x 10 columns

Este código recorre cada fila y si el año es anterior a 2009 y tiene un NaN, busca el valor del primer trimestre de ese año y lo reemplaza

3. Población y Cantidad de extranjeros

Las variables Población y Cantidad de Extranjeros presentan datos semestrales desde 2003 hasta 2021, mientras que el resto del dataset está en formato trimestral. Dada su evolución progresiva, se ha aplicado una interpolación lineal temporal para estimar los valores trimestrales de forma realista y coherente.

```
In [125... df_clean['Poblacion'] = df_clean.set_index('Fecha')['Poblacion'].interpolate(method='time').round().astype(int).df_clean['Cantidad de extranjeros'] = df_clean.set_index('Fecha')['Cantidad de extranjeros'].interpolate(method=df_clean.head(5))
```

Out[125...]

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjeros
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836	2362027
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	42011027	2503949
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231	2647431
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	42372802	2774802
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454	2900788

5 rows × 27 columns

4. Confianza del consumidor

Vamos a llenar los NaN de la variable Confianza_consumidor calculando las tasas de crecimiento trimestrales desde 2004 (año en el que empieza a haber datos) hasta 2007 (anterior a la crisis). Escogemos este período para que las tasas de crecimiento no se desvirtúen por el período posterior a la crisis, siguiendo la misma metodología aplicada para compraventas.

In [126...]

```
# con este codigo se me quedaban colgando los dos primeros trimestres de 2004, por lo que he tenido que hacer un df_24 = df_24.sort_values("Fecha").set_index("Fecha").asfreq("QE")  
  
# Calculamos la tasa de crecimiento trimestral (pct_change)  
df_24_crecimiento = df_24["Confianza_consumidor"].pct_change()  
  
# Calculamos el crecimiento medio solo entre 2004 y 2007  
crecimiento_medio = df_24_crecimiento["2004":"2007"].mean()  
  
primer_valor_2004 = df_24["Confianza_consumidor"]["2004-09-30"]  
  
# Calculamos valores 2003 aplicando crecimiento medio hacia atrás  
valores_2003 = [primer_valor_2004 / ((1 + crecimiento_medio) ** i) for i in range(1, 5)]  
fechas_2003 = pd.date_range(end="2003-12-31", periods=4, freq="QE")  
  
df_2003 = pd.DataFrame({  
    "Confianza_consumidor": valores_2003[::-1]  
}, index=fechas_2003)  
  
df_24 = pd.concat([df_2003, df_24]).sort_index().reset_index().rename(columns={"index": "Fecha"})
```

In [127...]

```
df_24 = df_24.sort_values("Fecha").set_index("Fecha").asfreq("QE")  
  
df_24_crecimiento = df_24["Confianza_consumidor"].pct_change()  
  
crecimiento_medio = df_24_crecimiento["2004":"2007"].mean()  
  
primeros_2004 = df_24["Confianza_consumidor"]["2004":].dropna()  
primer_valor_2004 = primeros_2004.iloc[0]  
fecha_valor_2004 = primeros_2004.index[0]  
  
num_trimestres = ((fecha_valor_2004.year - 2003) * 4 + (fecha_valor_2004.quarter - 1))  
  
valores = [primer_valor_2004 / ((1 + crecimiento_medio) ** i) for i in range(num_trimestres, 0, -1)]  
fechas = pd.date_range(start="2003-03-31", periods=num_trimestres, freq="QE")
```

```

df_fill = pd.DataFrame({
    "Confianza_consumidor": valores
}, index=fechas)

df_24 = df_24[~df_24.index.isin(fechas)]

df_24 = pd.concat([df_fill, df_24]).sort_index().reset_index().rename(columns={"index": "Fecha"})

```

```
/var/folders/cs/glwg8b695ggbr9k_ywkn40h40000gq/T/ipykernel_7142/2415504471.py:3: FutureWarning: The default fill_
method='pad' in Series.pct_change is deprecated and will be removed in a future version. Either fill in any non-
leading NA values prior to calling pct_change or specify 'fill_method=None' to not fill NA values.
    df_24_crecimiento = df_24["Confianza_consumidor"].pct_change()
```

In [128...]: df_clean = df_clean.merge(df_24, on="Fecha", how="outer")
df_clean.head(5)

Out[128...]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjeros
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836	2362027
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	42011027	2503949
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231	2647431
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	42372802	2774802
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454	2900788

5 rows × 28 columns

In [129...]: df_clean = df_clean.drop(columns=["Confianza_consumidor_x"])
df_clean = df_clean.rename(columns={"Confianza_consumidor_y": "Confianza del consumidor"})
df_clean.head(5)

Out[129...]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjeros
0	2003-03-31	263600	2.417559e+10	91713.17	4.693333	11.99	9568.0	4370.2	41827836	2362027
1	2003-06-30	247071	2.340044e+10	94711.40	4.453333	11.28	9701.0	4550.9	42011027	2503949
2	2003-09-30	236977	2.397833e+10	101184.20	4.213333	11.30	9816.4	4652.2	42196231	2647431
3	2003-12-31	241791	2.462120e+10	101828.44	3.883333	11.37	9858.9	4687.5	42372802	2774802
4	2004-03-31	283170	3.002192e+10	106020.83	3.760000	11.50	9914.9	4620.7	42547454	2900788

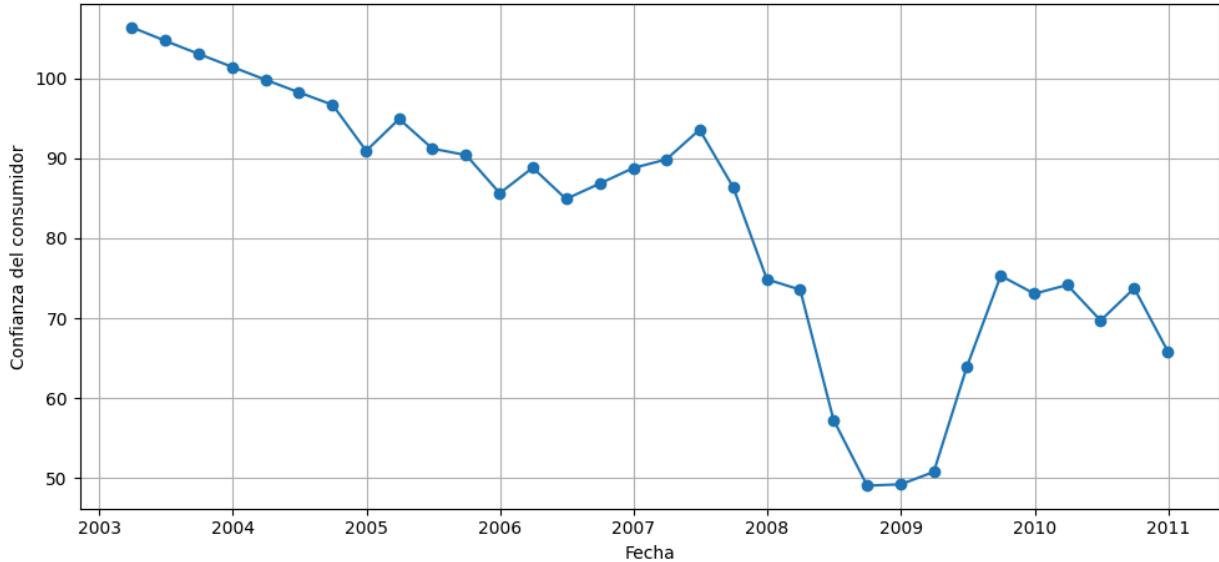
5 rows × 27 columns

In [130...]: import matplotlib.pyplot as plt

Filtrar Los datos entre 2003 y 2007
mask = (df_clean['Fecha'] >= '2003-01-01') & (df_clean['Fecha'] <= '2010-12-31')
df_plot = df_clean.loc[mask]

plt.figure(figsize=(10,5))
plt.plot(df_plot['Fecha'], df_plot['Confianza del consumidor'], marker='o', linestyle='--')
plt.title('Confianza del consumidor (2003-2007)')
plt.xlabel('Fecha')
plt.ylabel('Confianza del consumidor')
plt.grid(True)
plt.tight_layout()
plt.show()

Confianza del consumidor (2003-2007)



Descarga del Excel con todas las variables

```
In [131...]: from openpyxl import load_workbook
from openpyxl.utils import get_column_letter
from openpyxl.styles import numbers
from openpyxl.styles import PatternFill

# Guardar el DataFrame en Excel sin índice
nombre_archivo = "variables_limpias.xlsx"
df_clean.to_excel(nombre_archivo, sheet_name="variables", index=False)

# Ajustar el ancho de las columnas con openpyxl
# Cargar el archivo
wb = load_workbook(nombre_archivo)
ws = wb["variables"]

# Ajustar el ancho de cada columna según el contenido más Largo
for col_idx, column_cells in enumerate(ws.iter_cols(min_row=1, max_row=1), 1):
    max_length = 0
    for cell in column_cells:
        try:
            max_length = max(max_length, len(str(cell.value)))
        except:
            pass
    adjusted_width = max_length + 2 # Un poco de margen
    ws.column_dimensions[get_column_letter(col_idx)].width = adjusted_width

# Ajustar el ancho de la columna Fecha, al ser la primera es la "A"
ws.column_dimensions["A"].width = 20

# Aplicar formato de fecha corta a la columna A, si no, por defecto, Excel nos pone la fecha Larga incluyendo la
for cell in ws["A"][1:]: # Saltamos la cabecera
    cell.number_format = 'dd/mm/yyyy' # Este formato Excel lo traduce como "Fecha corta"

# Definir los colores
fill_header = PatternFill(start_color="A9D0F5", end_color="A9D0F5", fill_type="solid") # Azul claro para encabezado
fill_gray = PatternFill(start_color="D9D9D9", end_color="D9D9D9", fill_type="solid") # Gris claro
fill_white = PatternFill(start_color="FFFFFF", end_color="FFFFFF", fill_type="solid")

# Aplicar color al encabezado (primera fila)
for cell in ws[1]:
    cell.fill = fill_header

# Aplicar color por bloques de 4 filas
for i, row in enumerate(ws.iter_rows(min_row=2), start=0): # min_row=2 para saltar encabezado
    fill = fill_gray if (i // 4) % 2 == 0 else fill_white
    for cell in row:
        cell.fill = fill

# Guardar el archivo actualizado
wb.save(nombre_archivo)
print("Archivo variables_limpias.xlsx guardado y formateado correctamente")
```

Archivo variables_limpias.xlsx guardado y formateado correctamente

Análisis exploratorio de datos (EDA)

Tras la importación y revisión de las variables objetivo y explicativas que van a ser objeto de nuestro análisis, procedemos a realizar un análisis exploratorio y descriptivo inicial. El objetivo es entender el comportamiento de cada variable y las posibles relaciones que existan entre ellas.

0. Librerías, Dataset y configuraciones iniciales

Importamos el archivo excel resultante del anterior notebook donde ya tenemos todas las variables (objetivo y explicativas) correctamente importadas, limitadas al período temporal de nuestro análisis y con la periodicidad seleccionada (trimestral)

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy.stats import zscore
from scipy.stats import skew
from scipy.stats import boxcox
from scipy.stats.mstats import winsorize
archivo = "variables_limpias.xlsx"

# Dataframe con las variables Limpias y ordenadas trimestralmente
df_clean = pd.read_excel(archivo)
df_clean.head()
```

Out[1]:

	Fecha	Numero Hipotecas	Importe Hipotecas	valor_medio_hipotecas	Tipo interes Medio	Tasa Paro (%)	Contratos indefinidos	Contratos temporales	Poblacion	Cantidad de extranjeros
0	2003-03-31	263600	24175591000	91713.17	4.693333	11.99	9568.0	4370.2	41827836	2362027
1	2003-06-30	247071	23400441000	94711.40	4.453333	11.28	9701.0	4550.9	42011027	2503949
2	2003-09-30	236977	23978328000	101184.20	4.213333	11.30	9816.4	4652.2	42196231	2647431
3	2003-12-31	241791	24621201000	101828.44	3.883333	11.37	9858.9	4687.5	42372802	2774802
4	2004-03-31	283170	30021918000	106020.83	3.760000	11.50	9914.9	4620.7	42547454	2900788

5 rows × 27 columns

Tipo de datos de las variables

```
In [2]: df_clean.dtypes
```

```
Out[2]: Fecha                      datetime64[ns]
Numero Hipotecas                  int64
Importe Hipotecas                 int64
valor_medio_hipotecas            float64
Tipo interes Medio                float64
Tasa Paro (%)                     float64
Contratos indefinidos            float64
Contratos temporales             float64
Poblacion                         int64
Cantidad de extranjeros           int64
PIB Trimestral Ajustado          int64
PIB Trimestral No Ajustado       int64
IPI Variacion Anual Original    float64
IPI Variacion Anual Corregida   float64
Precio M2 Vivienda                float64
Euribor 12 Meses                  float64
MRO_Rate                           float64
TI_Credito                         float64
Renta disponible bruta           int64
IPC % Variación Anual            float64
Interes_Fijo_Hipotecas           float64
Interes_Variable_Hipotecas       float64
Afiliaciones_SS                   int64
Duracion_media_hipoteca          float64
Numero_compraventas              float64
Valor_compraventa                 float64
Confianza del consumidor          float64
dtype: object
```

Longitud del Data Frame

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

```
Out[3]: (84, 27)
```

Por tanto, tenemos 84 registros (datos trimestrales desde 2003 hasta 2023, incluidos) y 27 variables, siendo una de ellas la variable objetivo (Numero hipotecas)

Normalización del nombre de las variables

```
In [4]: df_clean.columns = [
    col.strip().title().replace(" ", "_") for col in df_clean.columns
]
df_clean.head()
```

```
Out[4]: Fecha  Numero_Hipotecas  Importe_Hipotecas  Valor_Medio_Hipotecas  Tipo_Interes_Medio  Tasa_Paro_(%)  Contratos_Inde
0  2003-03-31  263600  24175591000  91713.17  4.693333  11.99
1  2003-06-30  247071  23400441000  94711.40  4.453333  11.28
2  2003-09-30  236977  23978328000  101184.20  4.213333  11.30
3  2003-12-31  241791  24621201000  101828.44  3.883333  11.37
4  2004-03-31  283170  30021918000  106020.83  3.760000  11.50
```

5 rows × 27 columns

Valores nulos o faltantes

```
In [5]: null_counts_all = df_clean.isnull().sum().sort_values(ascending=False)
print("Valores nulos por columna:")
print(null_counts_all)
```

```
Valores nulos por columna:  
Fecha 0  
Numero_Hipotecas 0  
Importe_Hipotecas 0  
Valor_Medio_Hipotecas 0  
Tipo_Interes_Medio 0  
Tasa_Paro_(%) 0  
Contratos_Indefinidos 0  
Contratos_Temporales 0  
Poblacion 0  
Cantidad_De_Extranjeros 0  
Pib_Trimestral_Ajustado 0  
Pib_Trimestral_No_Ajustado 0  
Ipi_Variacion_Anual_Original 0  
Ipi_Variacion_Anual_Corregida 0  
Precio_M2_Vivienda 0  
Euribor_12_Meses 0  
Mro_Rate 0  
Ti_Credito 0  
Renta_Disponible_Bruta 0  
Ipc_%_Variación_Anual 0  
Interes_Fijo_Hipotecas 0  
Interes_Variable_Hipotecas 0  
Afiliaciones_Ss 0  
Duracion_Media_Hipoteca 0  
Numero_Compraventas 0  
Valor_Compraventa 0  
Confianza_Del_Consumidor 0  
dtype: int64
```

Vemos que no hay valores nulos, ya que este preprocesamiento lo hemos realizado en el anterior notebook a través de distintas técnicas explicadas en el mismo

1. Análisis descriptivo univariante

Realizamos el resumen estadístico de todas las variables numéricas. Este análisis es un punto de partida para entender los datos, ya que nos da una visión rápida de la calidad, rango, distribución y posibles problemas en las variables:

- Si el min o el max están muy alejados de la media o los cuartiles, puede haber outliers o errores en los datos
- Si la desviación estándar es muy alta, puede que tengamos que hacer alguna transformación en algún punto
- Si la media y la mediana son muy distintas, esto nos indica asimetría o sesgo en la variable
- Podemos ver qué variables tienen mayor variabilidad o valores extremos, lo que puede influir en decisiones de análisis

```
In [6]: desc = df_clean.drop(columns=['Fecha']).describe()  
print(desc)
```

Numero_Hipotecas	Importe_Hipotecas	Valor_Medio_Hipotecas	\	
count	84.000000	8.400000e+01	84.000000	
mean	146068.761905	1.826748e+10	121637.788452	
std	94770.421289	1.294296e+10	16144.326029	
min	41089.000000	4.048779e+09	91713.170000	
25%	75476.000000	8.548105e+09	107258.015000	
50%	102733.500000	1.357791e+10	120145.470000	
75%	233063.500000	2.402764e+10	137276.202500	
max	366094.000000	5.210989e+10	150998.360000	
Tipo_Interes_Medio	Tasa_Paro_(%)	Contratos_Indefinidos	\	
count	84.000000	84.000000	84.000000	
mean	3.458214	16.090952	11611.964286	
std	0.880841	5.476194	1140.069520	
min	1.800000	7.930000	9568.000000	
25%	2.760000	11.467500	10910.150000	
50%	3.486667	15.305000	11571.200000	
75%	4.091667	20.242500	12038.175000	
max	5.340000	26.940000	15123.100000	
Contratos_Temporales	Poblacion	Cantidad_De_Extranjeros	\	
count	84.000000	8.400000e+01	8.400000e+01	
mean	4180.639286	4.604990e+07	4.744000e+06	
std	701.287117	1.575921e+06	8.575177e+05	
min	3010.600000	4.182784e+07	2.362027e+06	
25%	3731.200000	4.578613e+07	4.417456e+06	
50%	4073.150000	4.650064e+07	4.900974e+06	
75%	4582.525000	4.681823e+07	5.335958e+06	
max	5717.000000	4.848686e+07	6.400849e+06	
Pib_Trimestral_Ajustado	...	Ti_Credito	Renta_Disponible_Bruta	\
count	84.000000	...	84.000000	84.000000
mean	276047.452381	...	2.973058	176242.357143
std	39138.601784	...	1.100019	28012.637617
min	195576.000000	...	1.431333	120199.000000
25%	257576.500000	...	2.014167	156062.750000
50%	269530.000000	...	2.989667	176178.000000
75%	297756.500000	...	3.486750	190131.500000
max	384835.000000	...	5.961667	258549.000000
Ipc_%_Variación_Anual	Interes_Fijo_Hipotecas	\		
count	84.000000	84.000000		
mean	2.224603	4.196667		
std	2.160502	1.273683		
min	-1.166667	1.910000		
25%	0.500000	3.170000		
50%	2.366667	4.330000		
75%	3.266667	5.270000		
max	10.066667	6.110000		
Interes_Variable_Hipotecas	Afiliaciones_Ss	Duracion_Media_Hipoteca	\	
count	84.000000	8.400000e+01	84.000000	
mean	3.377619	5.437575e+07	24.285833	
std	0.890017	3.592622e+06	1.419107	
min	1.900000	4.841314e+07	22.000000	
25%	2.760000	5.145852e+07	23.330000	
50%	3.560000	5.420097e+07	24.000000	
75%	4.080000	5.696682e+07	24.670000	
max	5.100000	6.238447e+07	28.000000	
Numero_Compraventas	Valor_Compraventa	Confianza_Del_Consumidor		
count	84.000000	8.400000e+01	84.000000	
mean	145815.083891	2.195423e+10	80.826471	
std	50292.510393	8.789130e+09	17.581999	
min	54835.000000	6.713009e+09	40.800000	
25%	107070.000000	1.491651e+10	69.325000	
50%	142066.000000	2.060044e+10	85.720000	
75%	184311.539978	2.810654e+10	95.095833	
max	251649.000000	4.294554e+10	106.736667	

[8 rows x 26 columns]

2. Distribución y forma de las variables

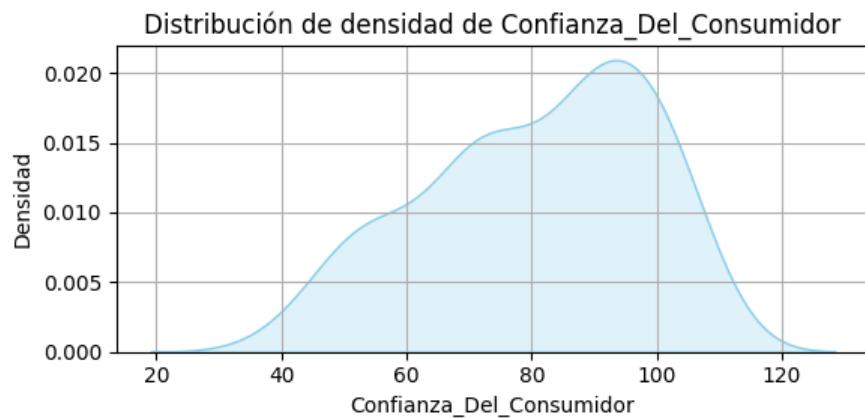
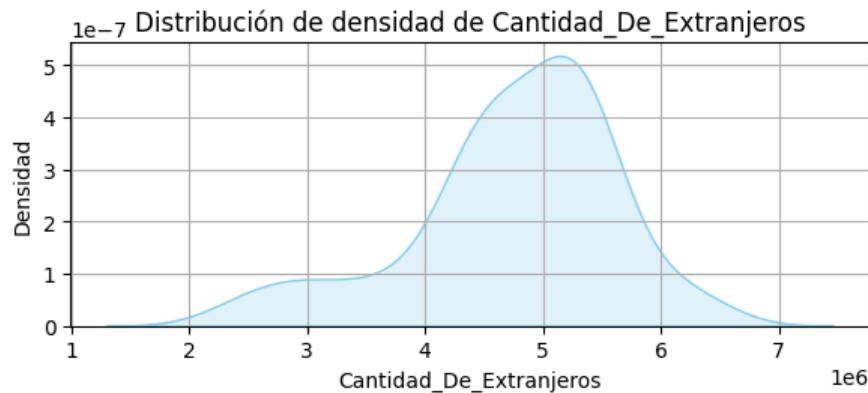
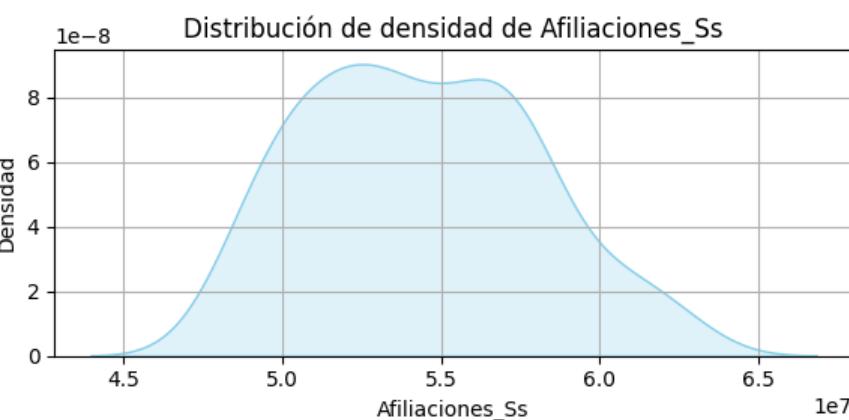
El objetivo de este apartado es comprobar si nuestras variables siguen o no una distribución normal, ya que esto va a condicionar análisis posteriores sobre las mismas. Es decir, nos va a ayudar a elegir qué métodos usar en siguientes apartados (por ejemplo, a la hora de estudiar la correlación entre variables o usar ciertos modelos económicos).

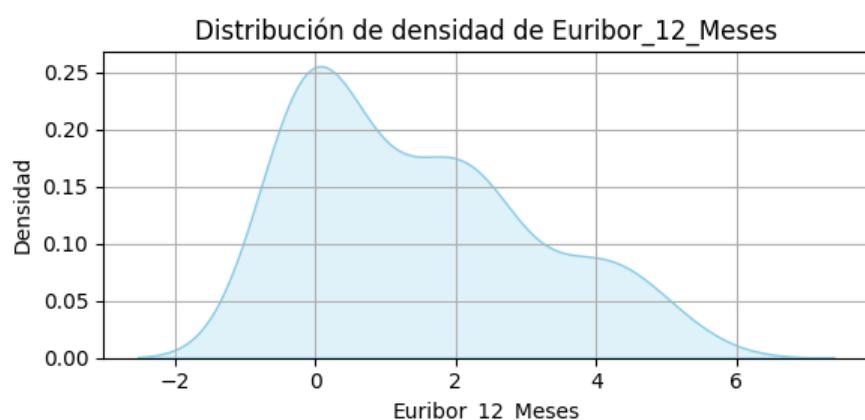
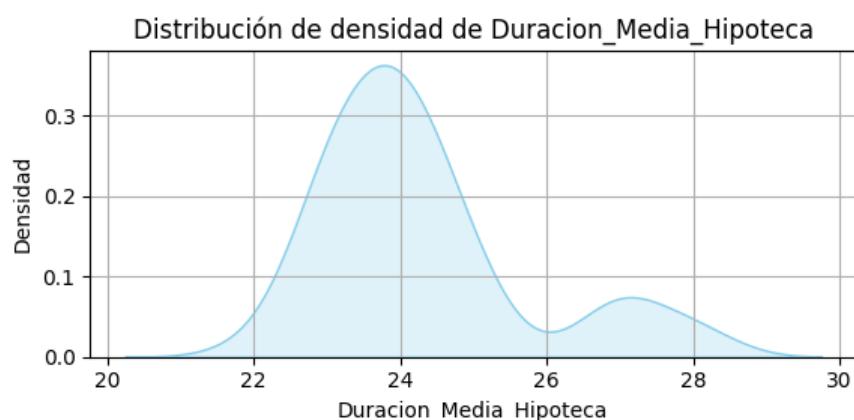
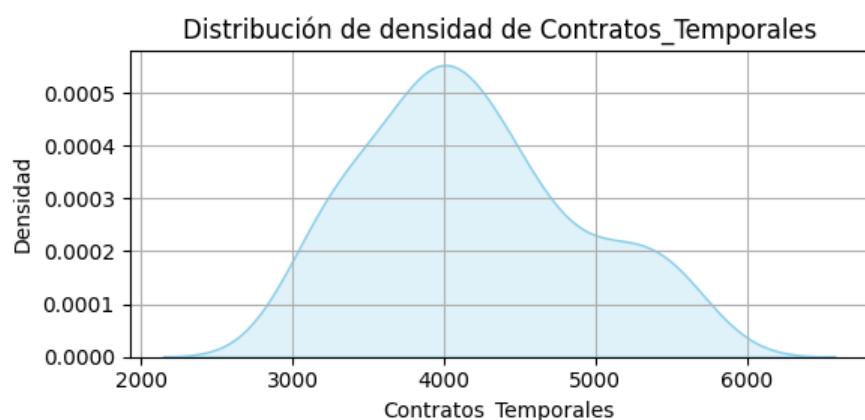
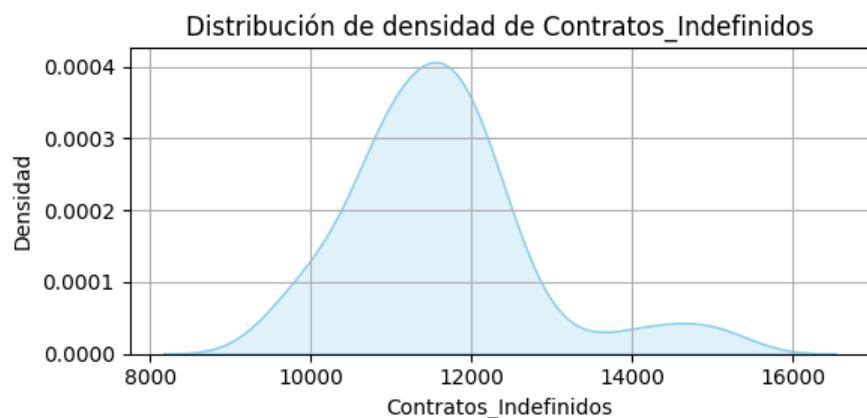
Gráficos de densidad

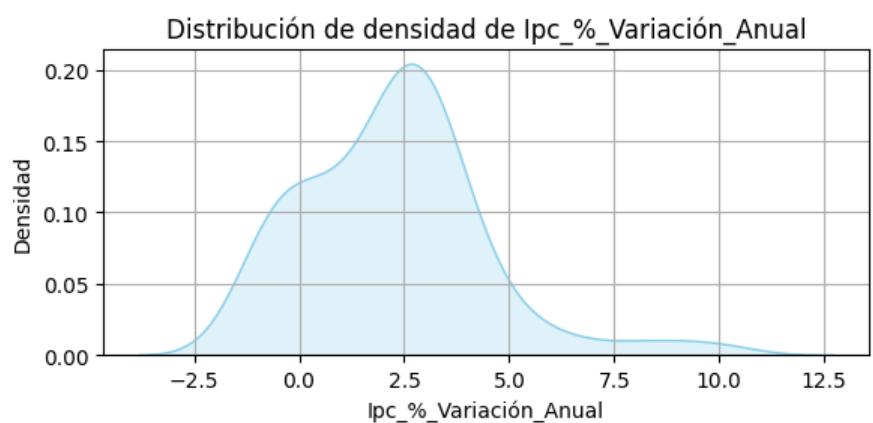
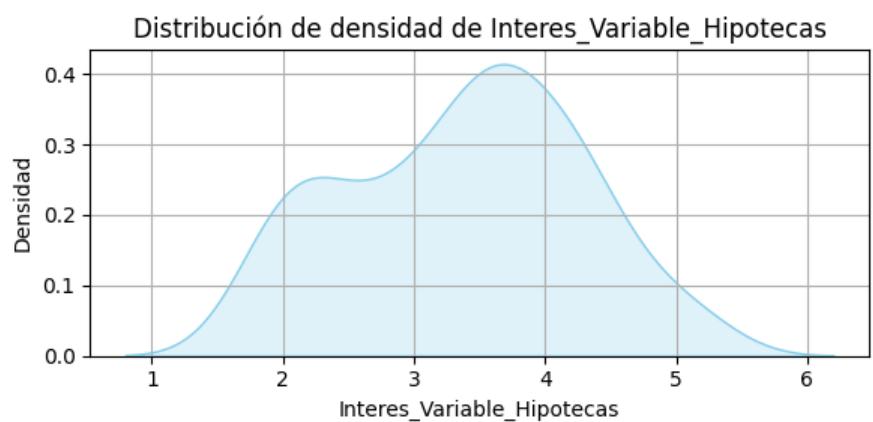
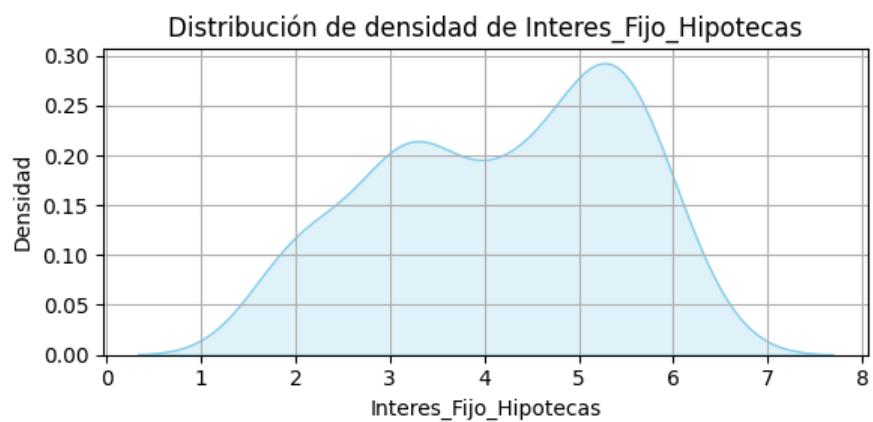
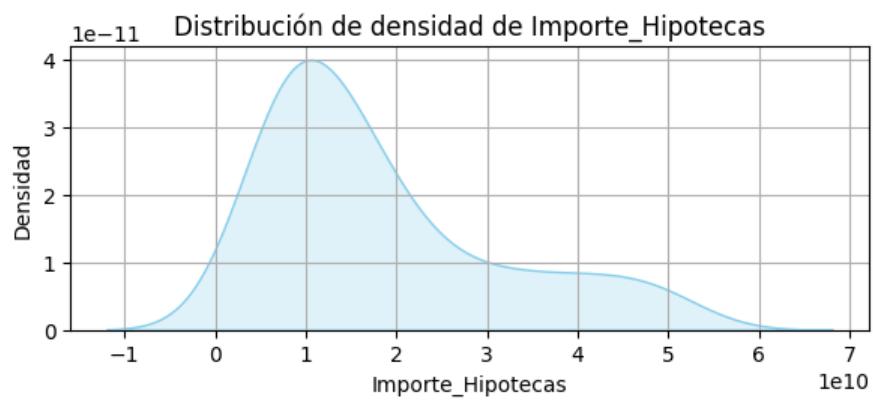
```
In [7]: cols = df_clean.columns.difference(['Fecha'])

for col in cols:
    plt.figure(figsize=(6, 3))
    sns.kdeplot(df_clean[col].dropna(), fill=True, color='skyblue')
    plt.title(f'Distribución de densidad de {col}', fontsize=12)
    plt.xlabel(col)
    plt.ylabel('Densidad')
    plt.grid(True)
    plt.tight_layout()
```

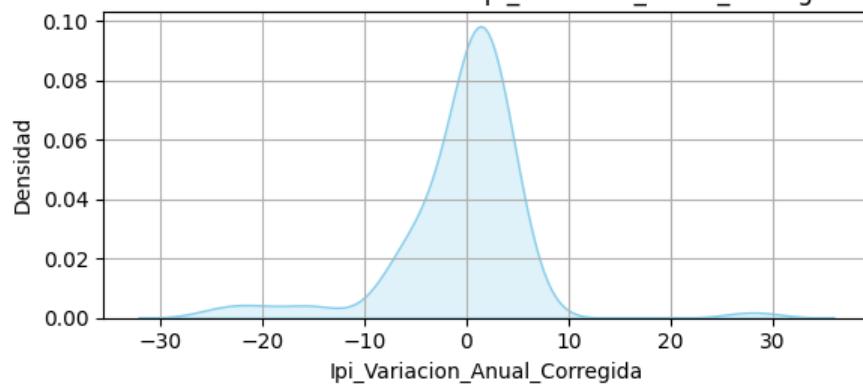
C:\Users\luife\AppData\Local\Temp\ipykernel_11708\1651398100.py:4: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`). Consider using `matplotlib.pyplot.close()`.



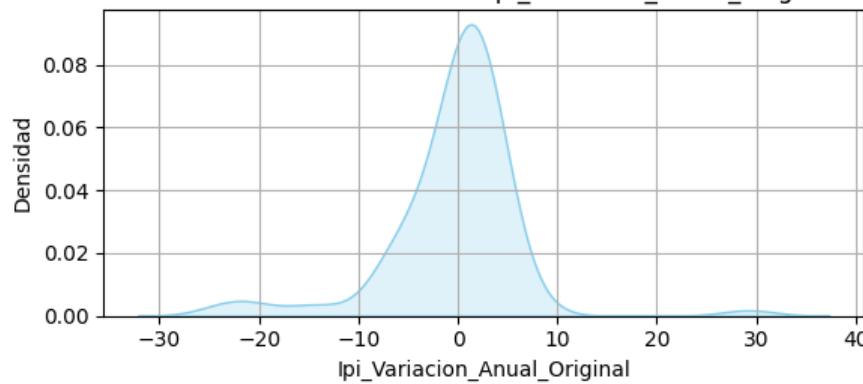




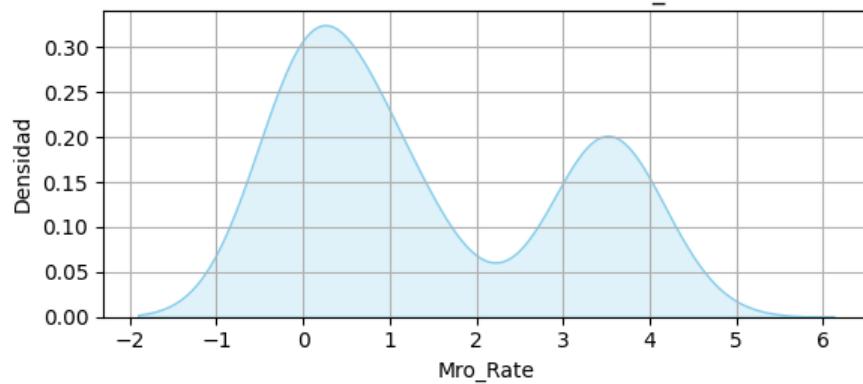
Distribución de densidad de Ipi_Variacion_Anual_Corregida



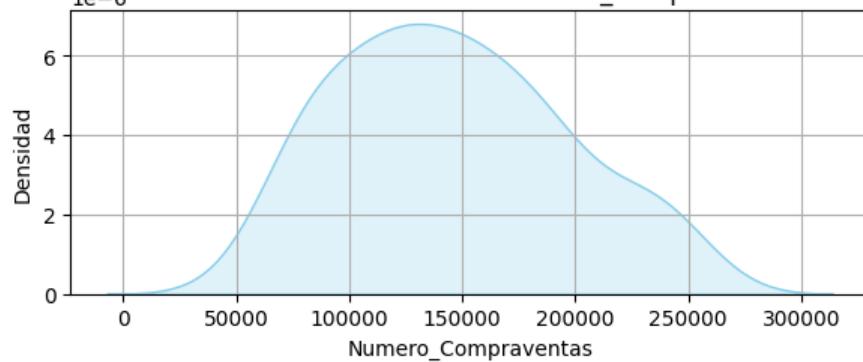
Distribución de densidad de Ipi_Variacion_Anual_Original

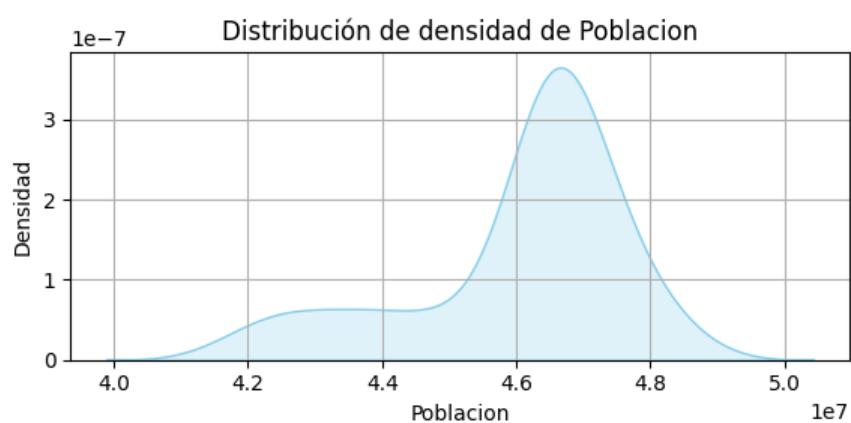
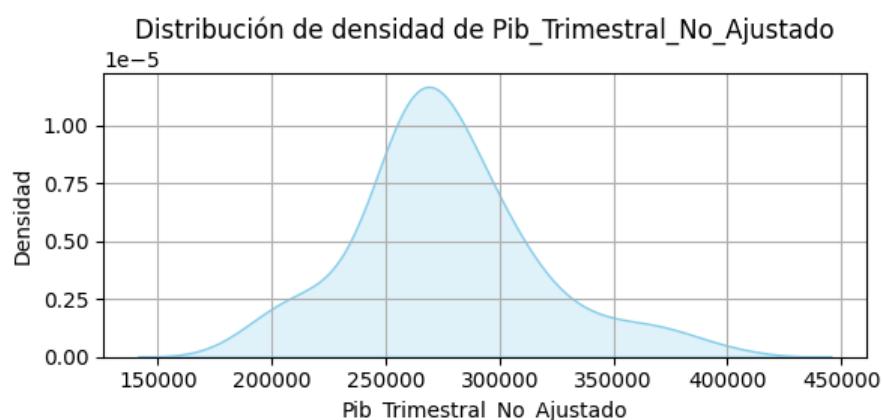
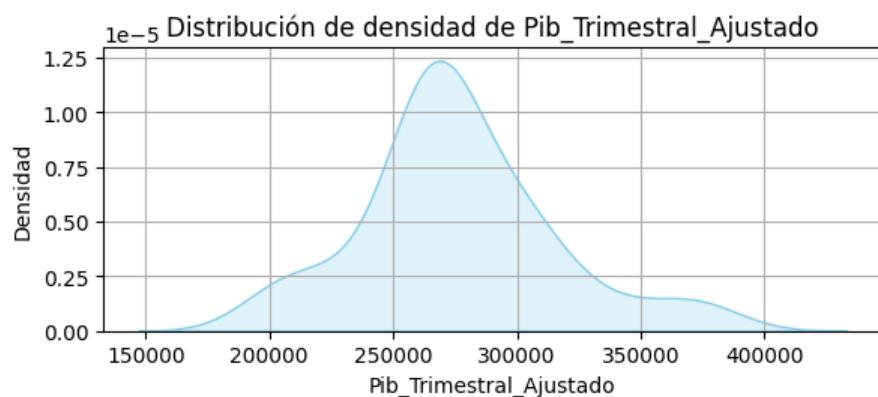
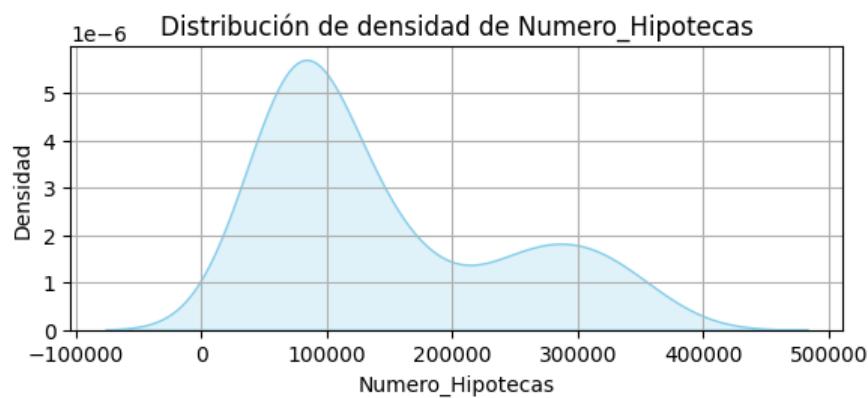


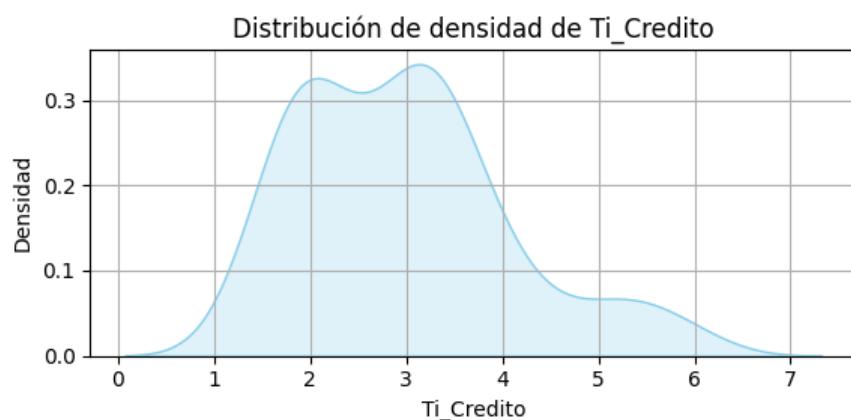
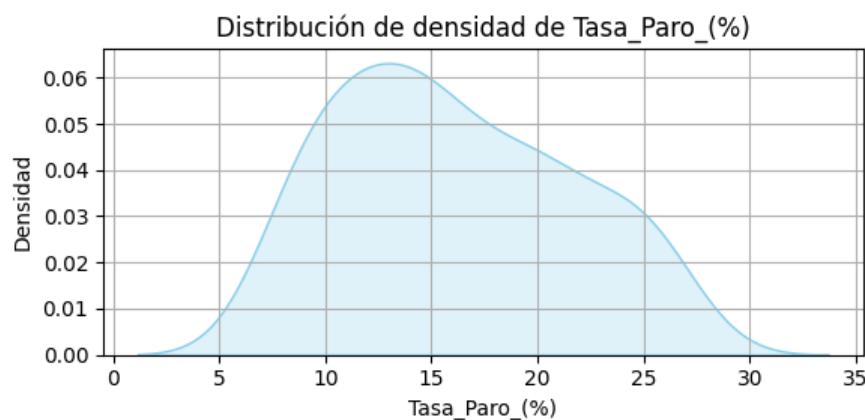
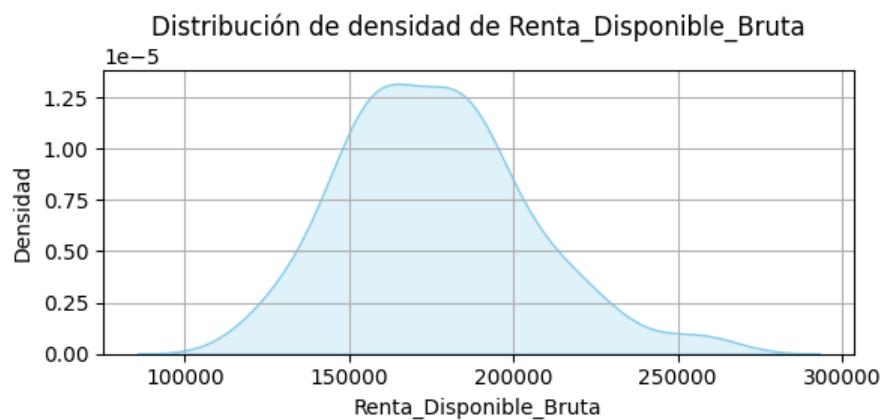
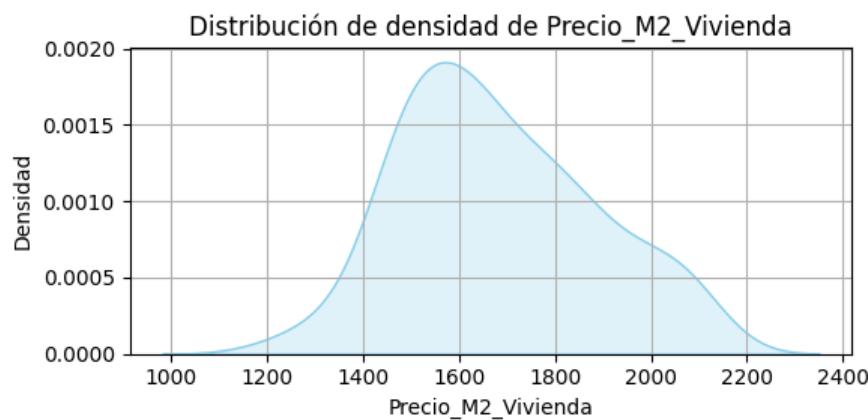
Distribución de densidad de Mro_Rate

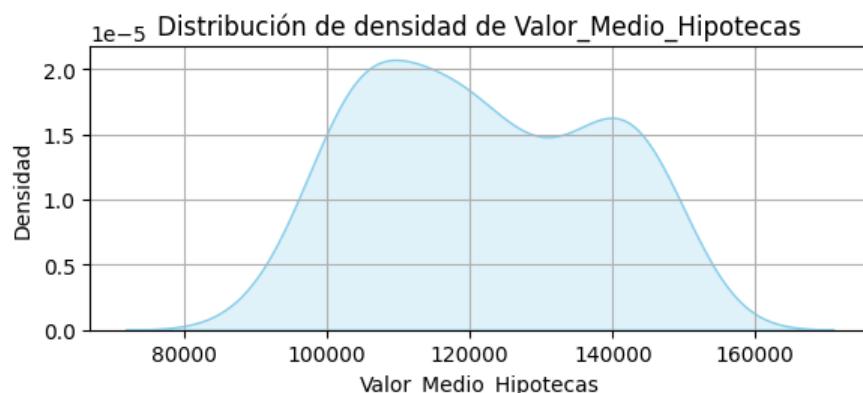
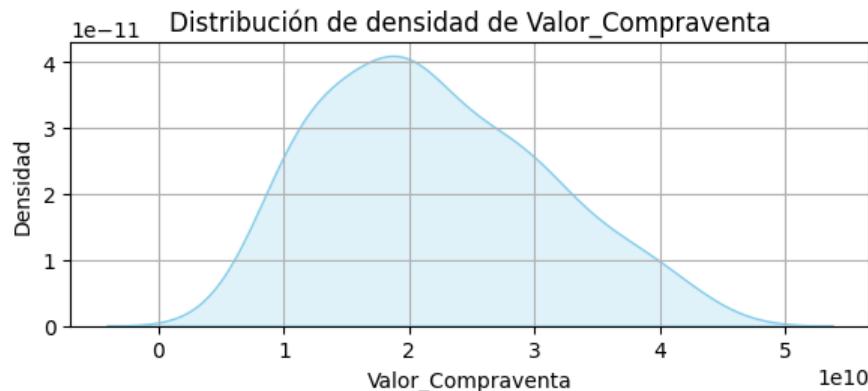
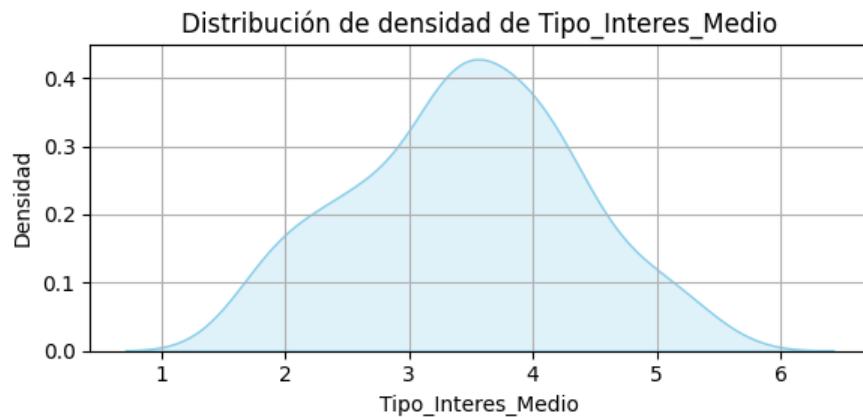


1e-6 Distribución de densidad de Numero_Compraventas









Tests de Normalidad

Realizamos distintos test de normalidad para comprobar si nuestras variables siguen o no una distribución normal. Los test seleccionados son Shapiro-Whilk, D'Agostino K2 y Anderson-Darling.

```
In [8]: def analizar_normalidad(df, excluir=['Fecha']):
    cols = df.select_dtypes(include=['float64', 'int64']).columns.difference(excluir)

    for col in cols:
        data = df[col].dropna()
        print(f"\n📊 Variable: {col}")

        if len(data) < 5000:
            stat_sw, p_sw = stats.shapiro(data)
            resultado_sw = "✅ Normal" if p_sw >= 0.05 else "❌ No normal"
            print(f" Shapiro-Wilk:    stat = {stat_sw:.4f}, p = {p_sw:.4f} → {resultado_sw}")
        else:
            print(" Shapiro-Wilk:    no aplicable (n > 5000)")

        stat_dag, p_dag = stats.normaltest(data)
        resultado_dag = "✅ Normal" if p_dag >= 0.05 else "❌ No normal"
        print(f" D'Agostino K2:   stat = {stat_dag:.4f}, p = {p_dag:.4f} → {resultado_dag}")

        ad_result = stats.anderson(data, dist='norm')
        print(f" Anderson-Darling: stat = {ad_result.statistic:.4f}")
        for i in range(len(ad_result.critical_values)):
            sl = ad_result.significance_level[i]
```

```

cv = ad_result.critical_values[i]
resultado = "✅ Normal" if ad_result.statistic < cv else "❌ No normal"
print(f"↳ {s1}: CV={cv:.3f} → {resultado}")

fig, axs = plt.subplots(1, 2, figsize=(12, 4))
sns.histplot(data, kde=True, ax=axs[0], color='skyblue')
axs[0].set_title(f"Histograma de {col}")

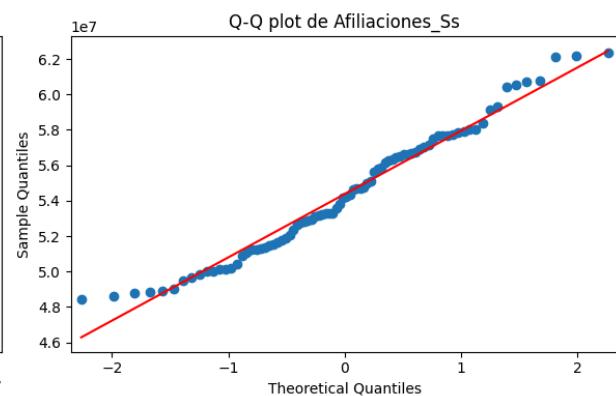
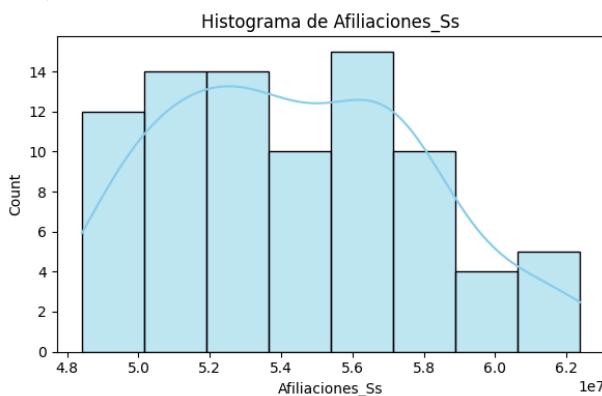
sm.qqplot(data, line='s', ax=axs[1])
axs[1].set_title(f"Q-Q plot de {col}")

plt.tight_layout()
plt.show()

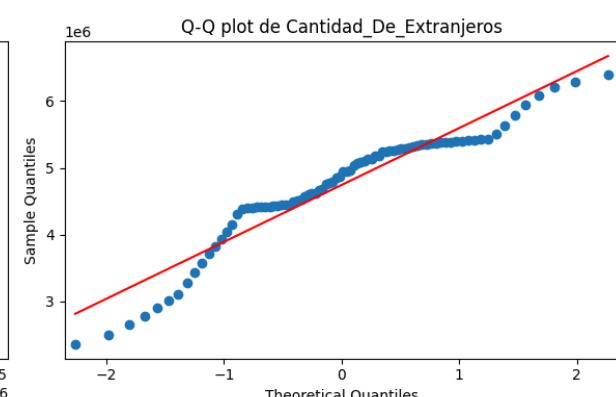
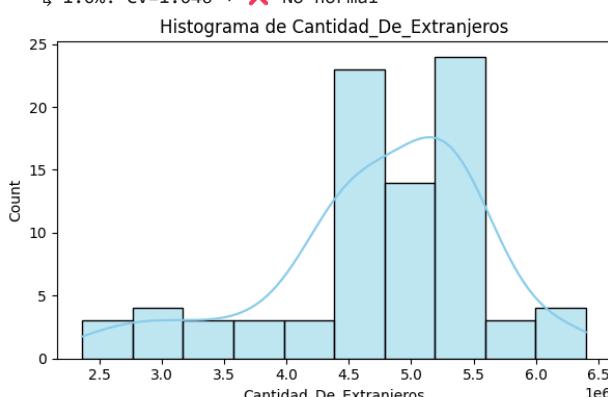
```

In [9]: `analizar_normalidad(df_clean)`

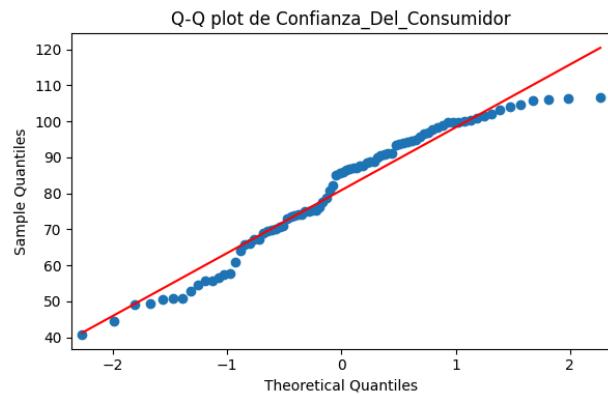
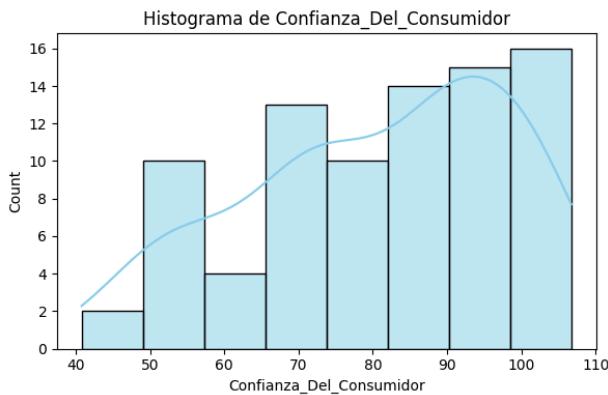
Variable: Afiliaciones_Ss
 Shapiro-Wilk: stat = 0.9692, p = 0.0416 → ❌ No normal
 D'Agostino K²: stat = 4.6065, p = 0.0999 → ✅ Normal
 Anderson-Darling: stat = 0.6138
 ↳ 15.0%: CV=0.552 → ❌ No normal
 ↳ 10.0%: CV=0.628 → ✅ Normal
 ↳ 5.0%: CV=0.754 → ✅ Normal
 ↳ 2.5%: CV=0.879 → ✅ Normal
 ↳ 1.0%: CV=1.046 → ✅ Normal



Variable: Cantidad_De_Extranjeros
 Shapiro-Wilk: stat = 0.9280, p = 0.0002 → ❌ No normal
 D'Agostino K²: stat = 11.3886, p = 0.0034 → ❌ No normal
 Anderson-Darling: stat = 2.1677
 ↳ 15.0%: CV=0.552 → ❌ No normal
 ↳ 10.0%: CV=0.628 → ❌ No normal
 ↳ 5.0%: CV=0.754 → ❌ No normal
 ↳ 2.5%: CV=0.879 → ❌ No normal
 ↳ 1.0%: CV=1.046 → ❌ No normal



Variable: Confianza_Del_Consumidor
 Shapiro-Wilk: stat = 0.9475, p = 0.0018 → ❌ No normal
 D'Agostino K²: stat = 9.4936, p = 0.0087 → ❌ No normal
 Anderson-Darling: stat = 1.3104
 ↳ 15.0%: CV=0.552 → ❌ No normal
 ↳ 10.0%: CV=0.628 → ❌ No normal
 ↳ 5.0%: CV=0.754 → ❌ No normal
 ↳ 2.5%: CV=0.879 → ❌ No normal
 ↳ 1.0%: CV=1.046 → ❌ No normal



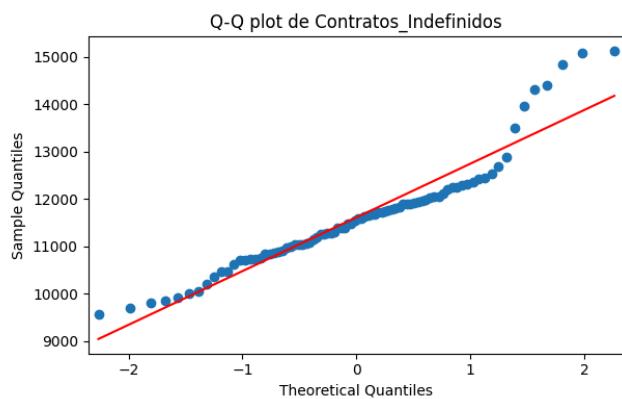
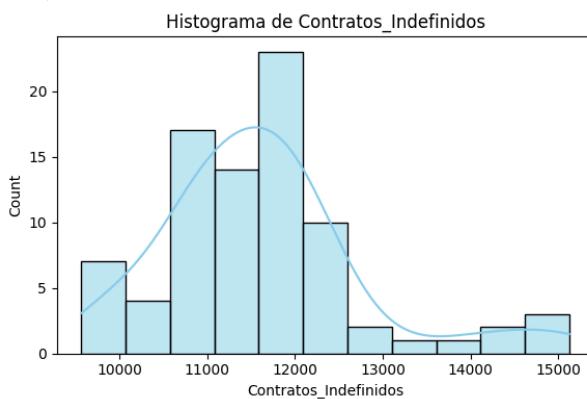
Variable: Contratos_Indefinidos

Shapiro-Wilk: stat = 0.9144, p = 0.0000 → X No normal

D'Agostino K²: stat = 20.1058, p = 0.0000 → X No normal

Anderson-Darling: stat = 1.9151

- ↳ 15.0%: CV=0.552 → X No normal
- ↳ 10.0%: CV=0.628 → X No normal
- ↳ 5.0%: CV=0.754 → X No normal
- ↳ 2.5%: CV=0.879 → X No normal
- ↳ 1.0%: CV=1.046 → X No normal



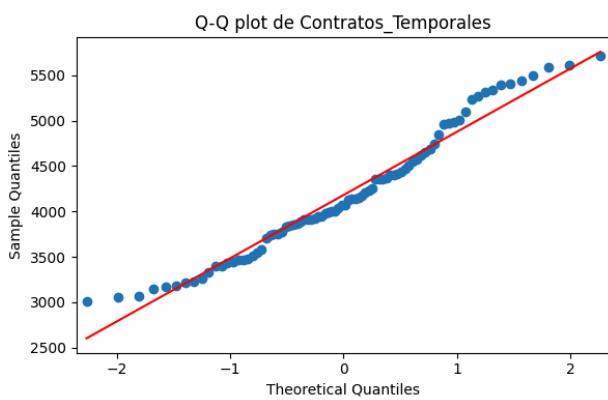
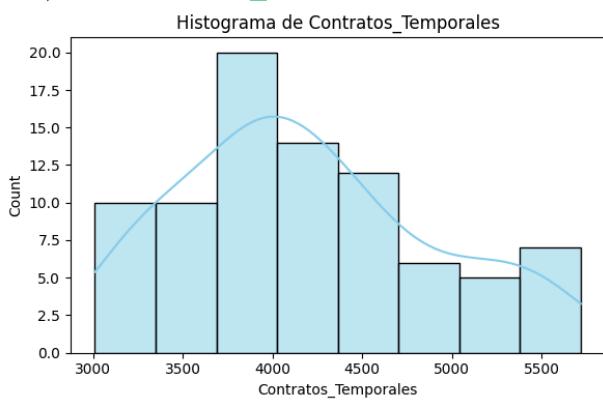
Variable: Contratos_Temporales

Shapiro-Wilk: stat = 0.9616, p = 0.0133 → X No normal

D'Agostino K²: stat = 4.7185, p = 0.0945 → ✓ Normal

Anderson-Darling: stat = 0.8481

- ↳ 15.0%: CV=0.552 → X No normal
- ↳ 10.0%: CV=0.628 → X No normal
- ↳ 5.0%: CV=0.754 → X No normal
- ↳ 2.5%: CV=0.879 → ✓ Normal
- ↳ 1.0%: CV=1.046 → ✓ Normal



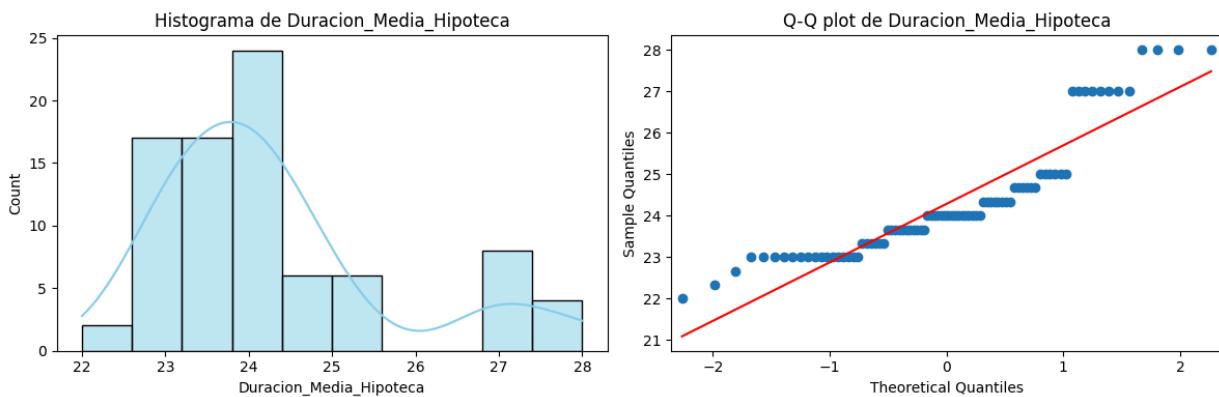
Variable: Duracion_Media_Hipoteca

Shapiro-Wilk: stat = 0.8373, p = 0.0000 → X No normal

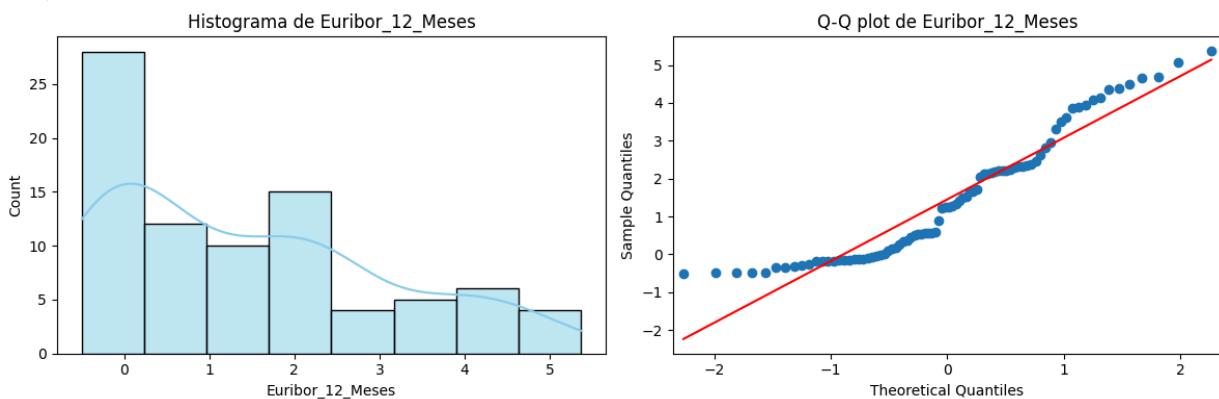
D'Agostino K²: stat = 20.5230, p = 0.0000 → X No normal

Anderson-Darling: stat = 5.0666

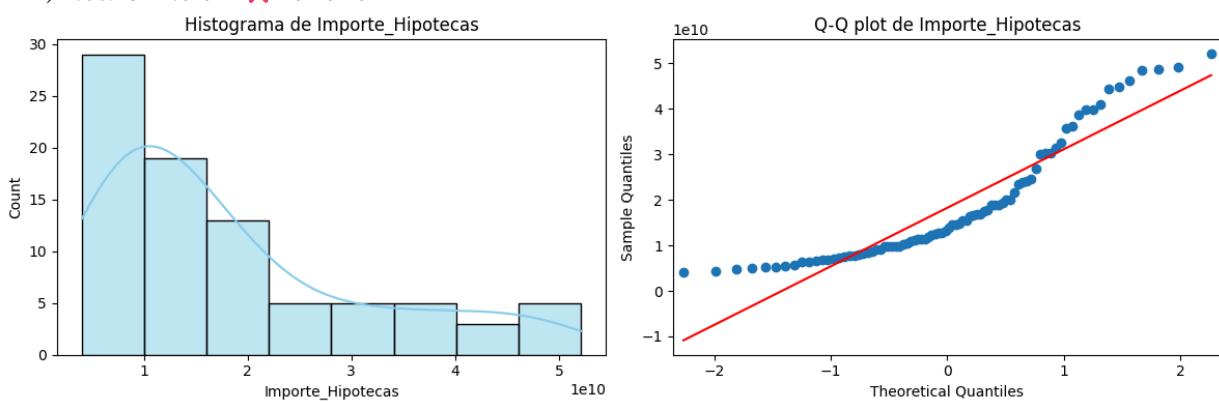
- ↳ 15.0%: CV=0.552 → X No normal
- ↳ 10.0%: CV=0.628 → X No normal
- ↳ 5.0%: CV=0.754 → X No normal
- ↳ 2.5%: CV=0.879 → X No normal
- ↳ 1.0%: CV=1.046 → X No normal



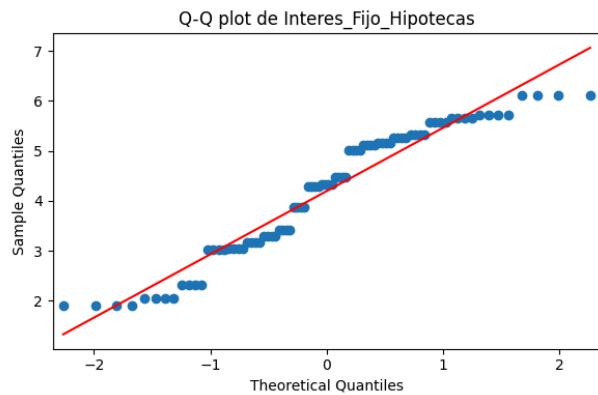
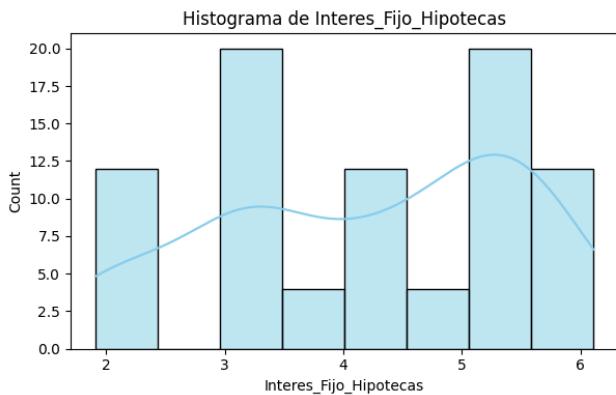
Variable: Euribor_12_Meses
 Shapiro-Wilk: stat = 0.9059, p = 0.0000 → No normal
 D'Agostino K²: stat = 8.7243, p = 0.0128 → No normal
 Anderson-Darling: stat = 2.5377
 ↴ 15.0%: CV=0.552 → No normal
 ↴ 10.0%: CV=0.628 → No normal
 ↴ 5.0%: CV=0.754 → No normal
 ↴ 2.5%: CV=0.879 → No normal
 ↴ 1.0%: CV=1.046 → No normal



Variable: Importe_Hipotecas
 Shapiro-Wilk: stat = 0.8520, p = 0.0000 → No normal
 D'Agostino K²: stat = 14.8712, p = 0.0006 → No normal
 Anderson-Darling: stat = 4.4197
 ↴ 15.0%: CV=0.552 → No normal
 ↴ 10.0%: CV=0.628 → No normal
 ↴ 5.0%: CV=0.754 → No normal
 ↴ 2.5%: CV=0.879 → No normal
 ↴ 1.0%: CV=1.046 → No normal



Variable: Interes_Fijo_Hipotecas
 Shapiro-Wilk: stat = 0.9261, p = 0.0001 → No normal
 D'Agostino K²: stat = 23.0493, p = 0.0000 → No normal
 Anderson-Darling: stat = 2.0486
 ↴ 15.0%: CV=0.552 → No normal
 ↴ 10.0%: CV=0.628 → No normal
 ↴ 5.0%: CV=0.754 → No normal
 ↴ 2.5%: CV=0.879 → No normal
 ↴ 1.0%: CV=1.046 → No normal



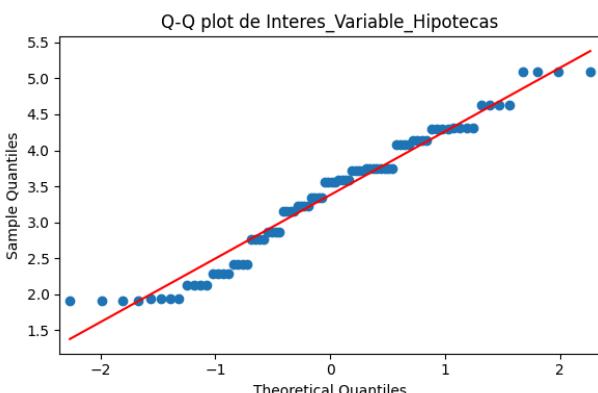
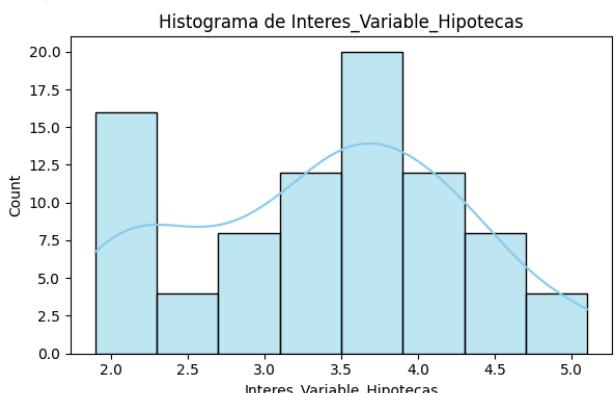
Variable: Interes_Variable_Hipotecas

Shapiro-Wilk: stat = 0.9568, p = 0.0066 → X No normal

D'Agostino K²: stat = 6.5733, p = 0.0374 → X No normal

Anderson-Darling: stat = 0.9976

- ↳ 15.0%: CV=0.552 → X No normal
- ↳ 10.0%: CV=0.628 → X No normal
- ↳ 5.0%: CV=0.754 → X No normal
- ↳ 2.5%: CV=0.879 → X No normal
- ↳ 1.0%: CV=1.046 → ✓ Normal



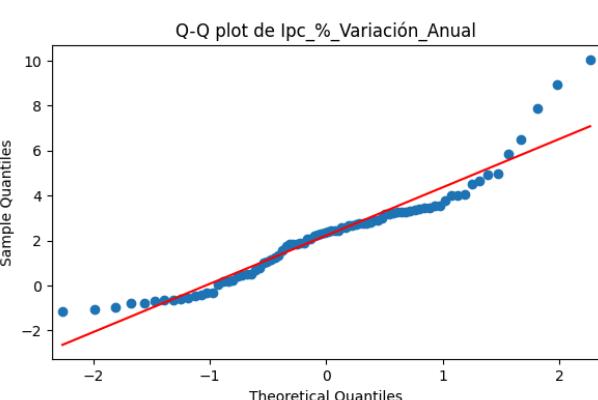
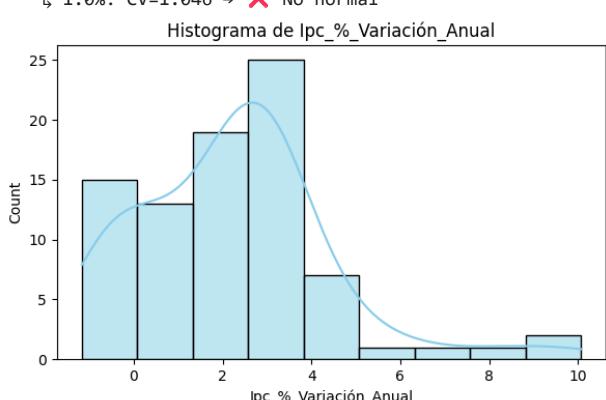
Variable: Ipc_%_Variación_Annual

Shapiro-Wilk: stat = 0.9283, p = 0.0002 → X No normal

D'Agostino K²: stat = 17.8637, p = 0.0001 → X No normal

Anderson-Darling: stat = 1.1989

- ↳ 15.0%: CV=0.552 → X No normal
- ↳ 10.0%: CV=0.628 → X No normal
- ↳ 5.0%: CV=0.754 → X No normal
- ↳ 2.5%: CV=0.879 → X No normal
- ↳ 1.0%: CV=1.046 → X No normal



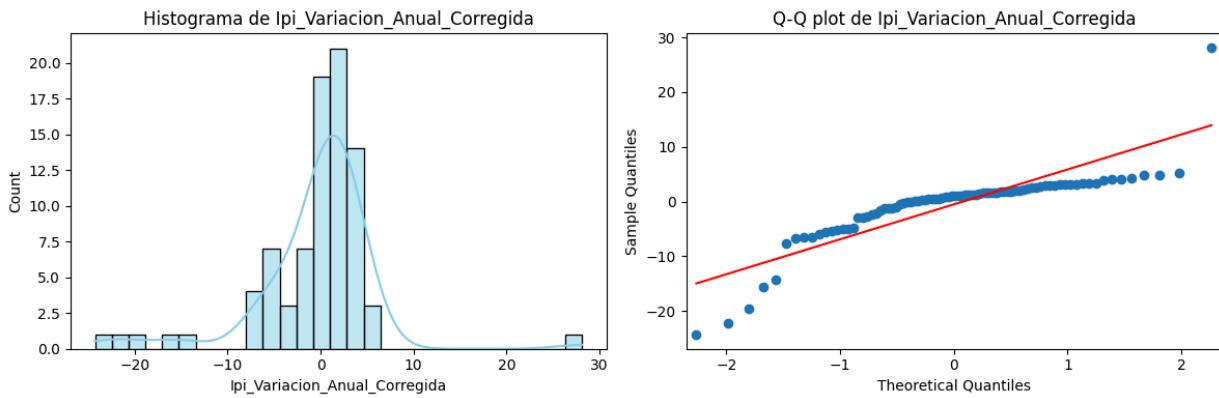
Variable: Ipi_Variacion_Annual_Corregida

Shapiro-Wilk: stat = 0.7533, p = 0.0000 → X No normal

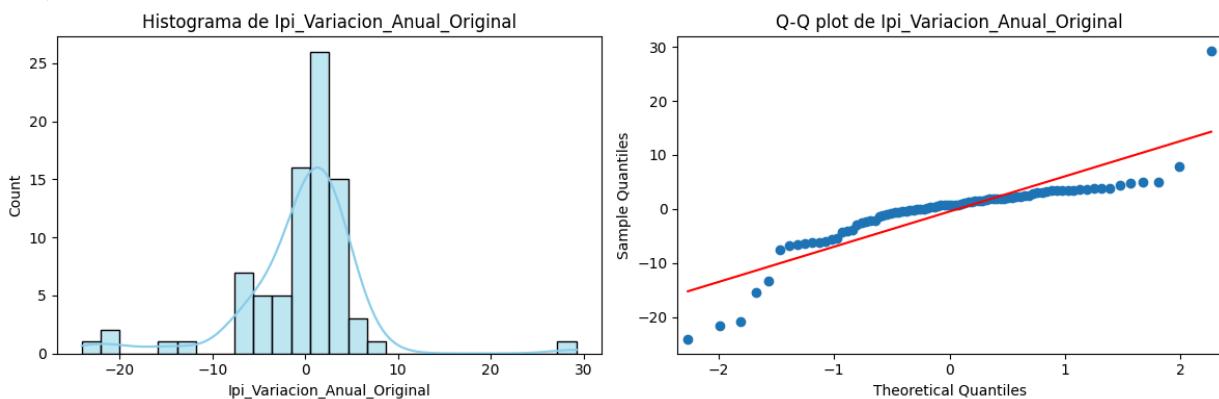
D'Agostino K²: stat = 27.8272, p = 0.0000 → X No normal

Anderson-Darling: stat = 6.5532

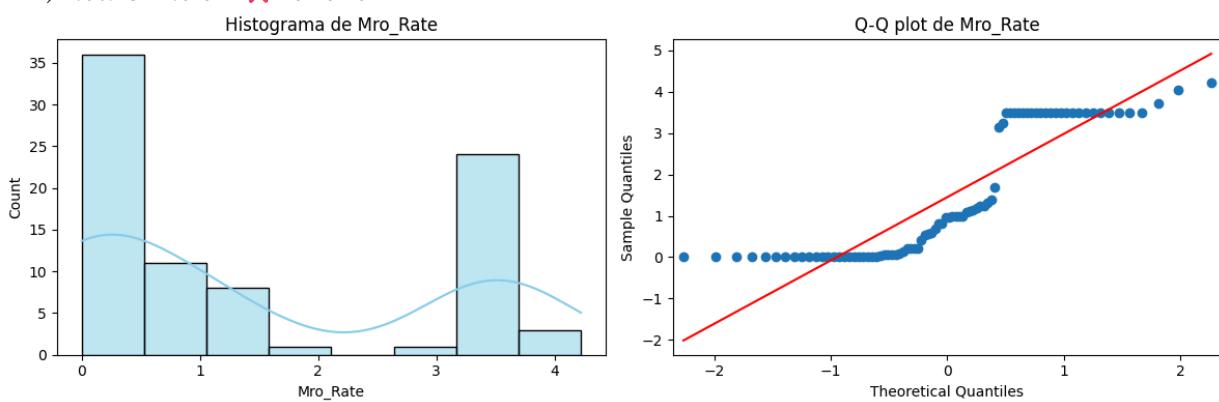
- ↳ 15.0%: CV=0.552 → X No normal
- ↳ 10.0%: CV=0.628 → X No normal
- ↳ 5.0%: CV=0.754 → X No normal
- ↳ 2.5%: CV=0.879 → X No normal
- ↳ 1.0%: CV=1.046 → X No normal



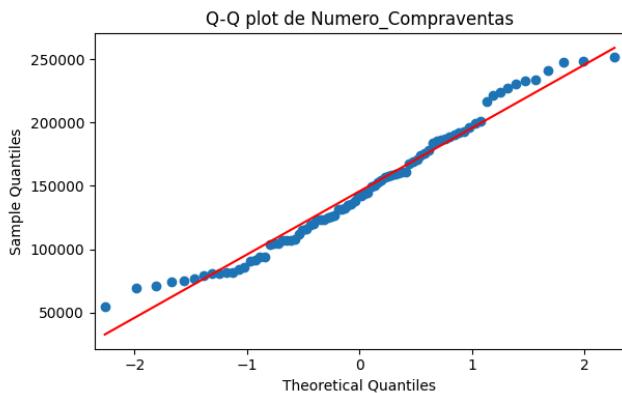
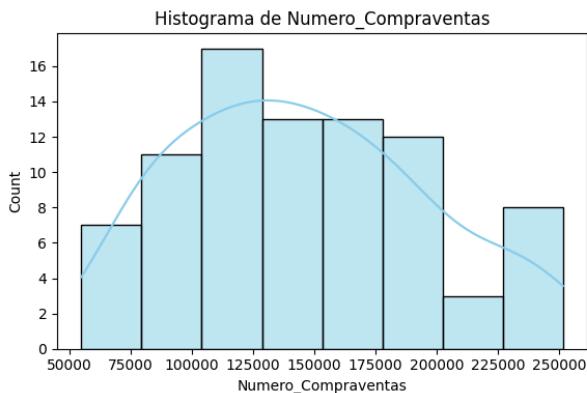
Variable: Ipi_Variacion_Anual_Original
 Shapiro-Wilk: stat = 0.7764, p = 0.0000 → ✗ No normal
 D'Agostino K²: stat = 25.2078, p = 0.0000 → ✗ No normal
 Anderson-Darling: stat = 5.6820
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✗ No normal
 ↴ 5.0%: CV=0.754 → ✗ No normal
 ↴ 2.5%: CV=0.879 → ✗ No normal
 ↴ 1.0%: CV=1.046 → ✗ No normal



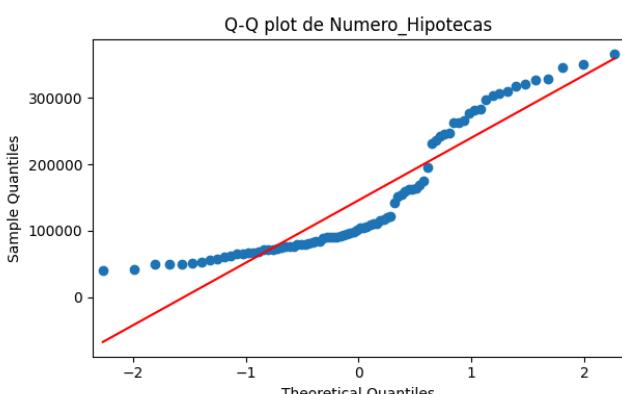
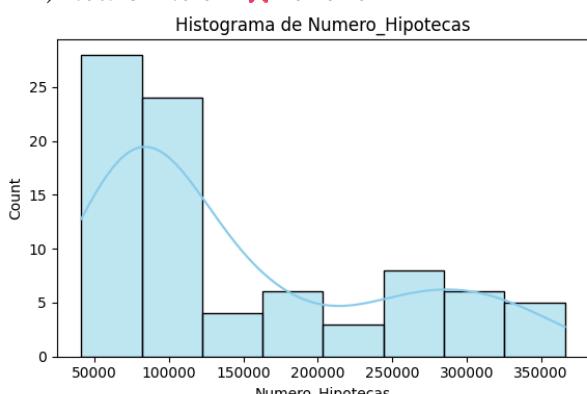
Variable: Mro_Rate
 Shapiro-Wilk: stat = 0.7699, p = 0.0000 → ✗ No normal
 D'Agostino K²: stat = 132.3652, p = 0.0000 → ✗ No normal
 Anderson-Darling: stat = 8.2035
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✗ No normal
 ↴ 5.0%: CV=0.754 → ✗ No normal
 ↴ 2.5%: CV=0.879 → ✗ No normal
 ↴ 1.0%: CV=1.046 → ✗ No normal



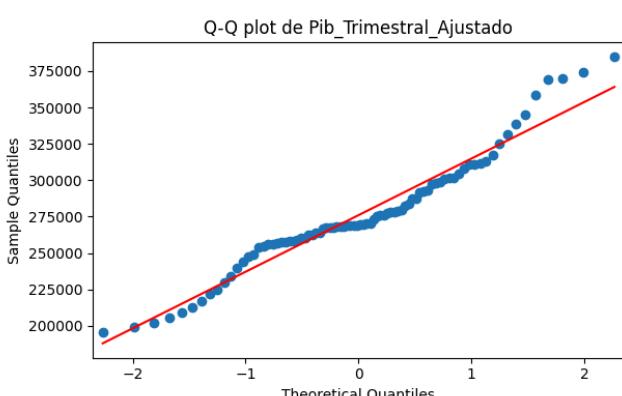
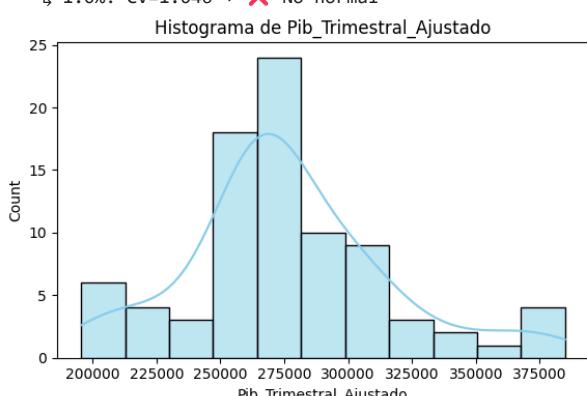
Variable: Numero_Compraventas
 Shapiro-Wilk: stat = 0.9673, p = 0.0310 → ✗ No normal
 D'Agostino K²: stat = 5.6084, p = 0.0606 → ✓ Normal
 Anderson-Darling: stat = 0.6258
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✓ Normal
 ↴ 5.0%: CV=0.754 → ✓ Normal
 ↴ 2.5%: CV=0.879 → ✓ Normal
 ↴ 1.0%: CV=1.046 → ✓ Normal



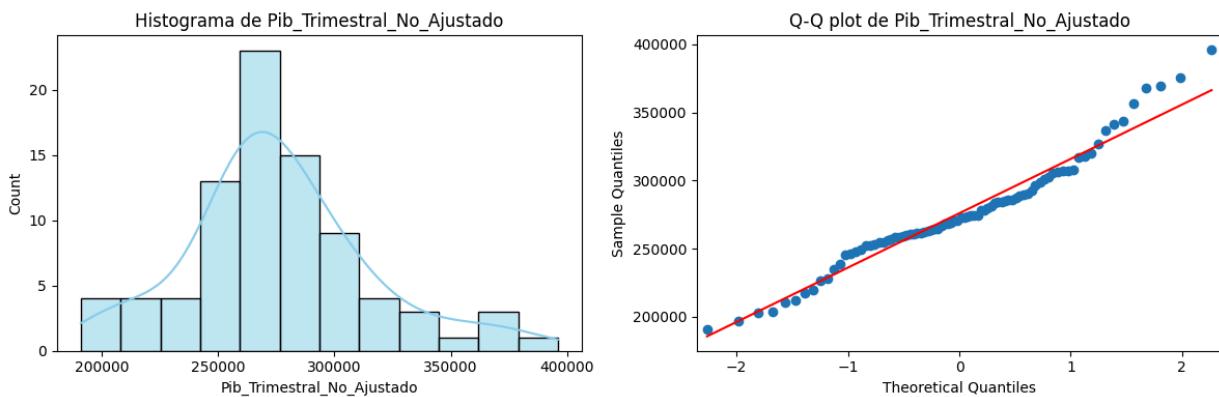
Variable: Numero_Hipotecas
 Shapiro-Wilk: stat = 0.8384, p = 0.0000 → X No normal
 D'Agostino K²: stat = 12.2933, p = 0.0021 → X No normal
 Anderson-Darling: stat = 5.5238
 ↴ 15.0%: CV=0.552 → X No normal
 ↴ 10.0%: CV=0.628 → X No normal
 ↴ 5.0%: CV=0.754 → X No normal
 ↴ 2.5%: CV=0.879 → X No normal
 ↴ 1.0%: CV=1.046 → X No normal



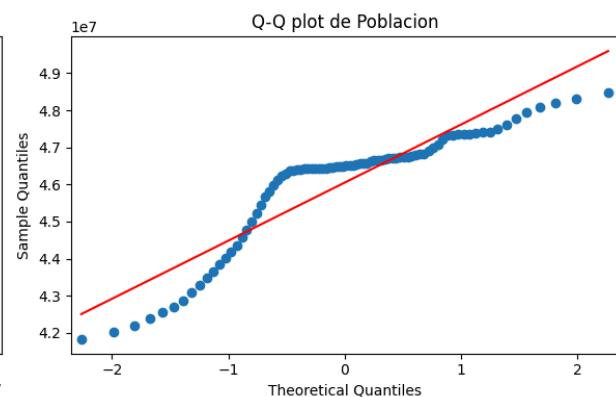
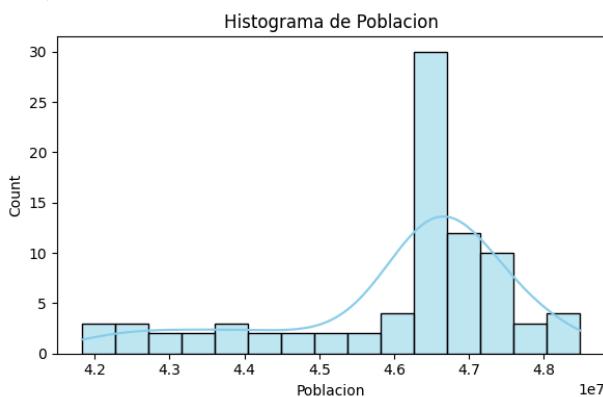
Variable: Pib_Trimestral_Ajustado
 Shapiro-Wilk: stat = 0.9548, p = 0.0050 → X No normal
 D'Agostino K²: stat = 5.9138, p = 0.0520 → ✓ Normal
 Anderson-Darling: stat = 1.4224
 ↴ 15.0%: CV=0.552 → X No normal
 ↴ 10.0%: CV=0.628 → X No normal
 ↴ 5.0%: CV=0.754 → X No normal
 ↴ 2.5%: CV=0.879 → X No normal
 ↴ 1.0%: CV=1.046 → X No normal



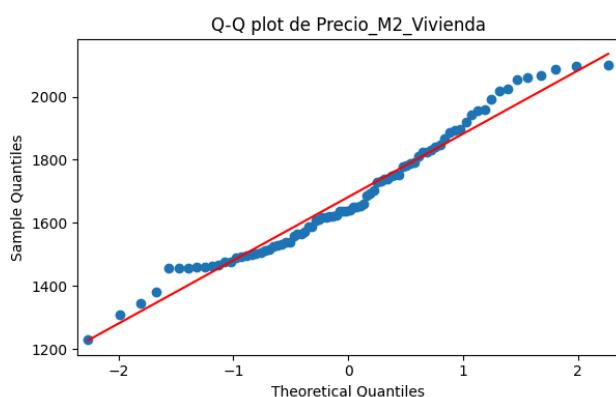
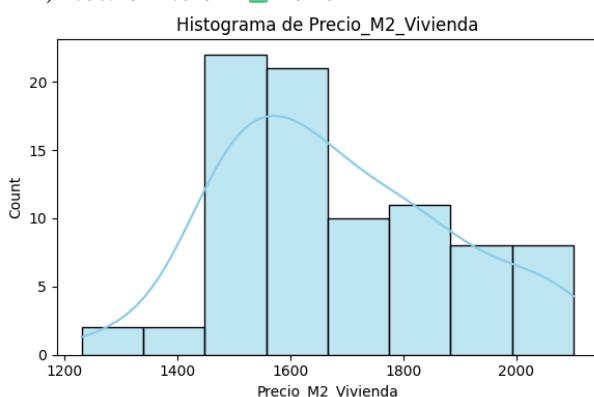
Variable: Pib_Trimestral_No_Ajustado
 Shapiro-Wilk: stat = 0.9631, p = 0.0166 → X No normal
 D'Agostino K²: stat = 6.5796, p = 0.0373 → X No normal
 Anderson-Darling: stat = 1.1242
 ↴ 15.0%: CV=0.552 → X No normal
 ↴ 10.0%: CV=0.628 → X No normal
 ↴ 5.0%: CV=0.754 → X No normal
 ↴ 2.5%: CV=0.879 → X No normal
 ↴ 1.0%: CV=1.046 → X No normal



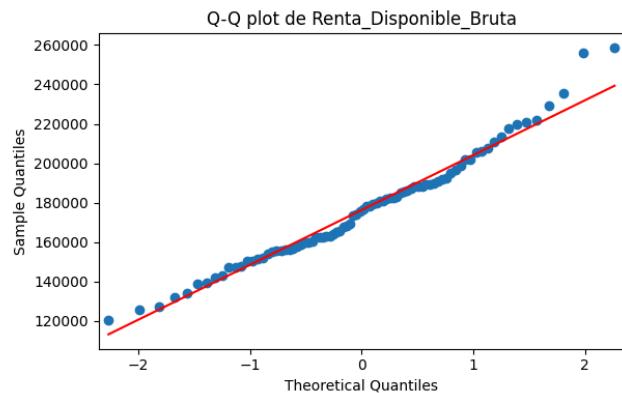
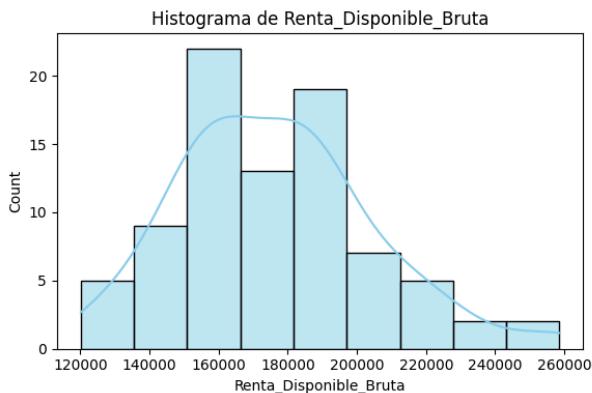
Variable: Poblacion
 Shapiro-Wilk: stat = 0.8459, p = 0.0000 → X No normal
 D'Agostino K²: stat = 17.5962, p = 0.0002 → X No normal
 Anderson-Darling: stat = 5.6426
 ↴ 15.0%: CV=0.552 → X No normal
 ↴ 10.0%: CV=0.628 → X No normal
 ↴ 5.0%: CV=0.754 → X No normal
 ↴ 2.5%: CV=0.879 → X No normal
 ↴ 1.0%: CV=1.046 → X No normal



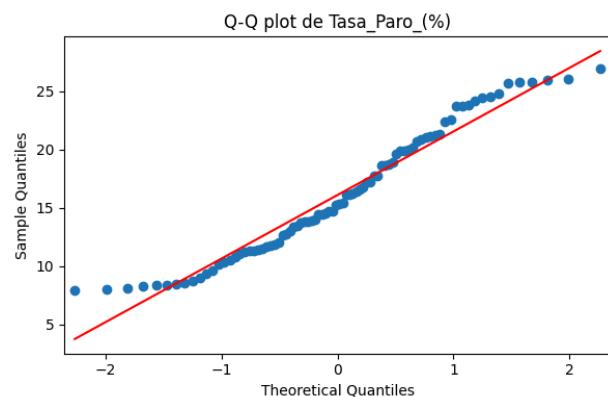
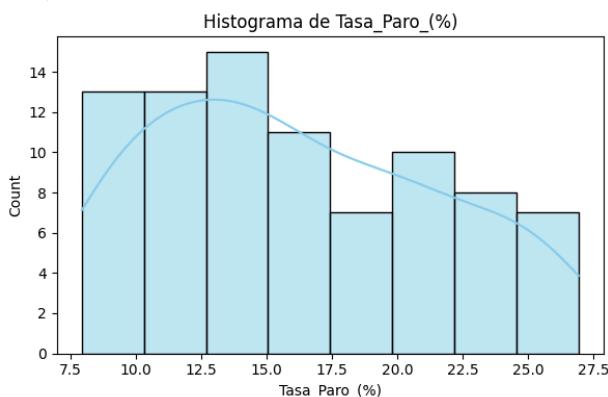
Variable: Precio_M2_Vivienda
 Shapiro-Wilk: stat = 0.9668, p = 0.0290 → X No normal
 D'Agostino K²: stat = 3.3462, p = 0.1877 → ✓ Normal
 Anderson-Darling: stat = 0.9461
 ↴ 15.0%: CV=0.552 → X No normal
 ↴ 10.0%: CV=0.628 → X No normal
 ↴ 5.0%: CV=0.754 → X No normal
 ↴ 2.5%: CV=0.879 → X No normal
 ↴ 1.0%: CV=1.046 → ✓ Normal



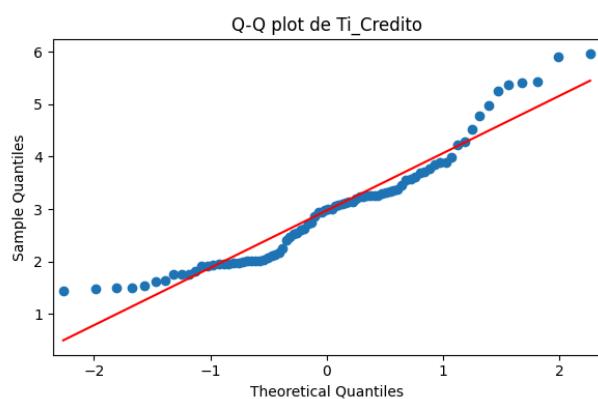
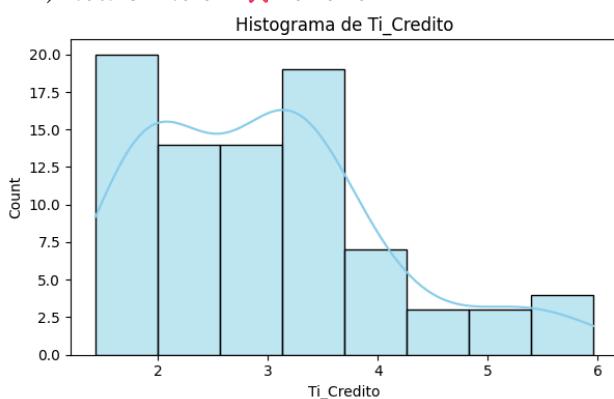
Variable: Renta_Disponible_Bruta
 Shapiro-Wilk: stat = 0.9758, p = 0.1142 → ✓ Normal
 D'Agostino K²: stat = 5.5239, p = 0.0632 → ✓ Normal
 Anderson-Darling: stat = 0.5089
 ↴ 15.0%: CV=0.552 → ✓ Normal
 ↴ 10.0%: CV=0.628 → ✓ Normal
 ↴ 5.0%: CV=0.754 → ✓ Normal
 ↴ 2.5%: CV=0.879 → ✓ Normal
 ↴ 1.0%: CV=1.046 → ✓ Normal



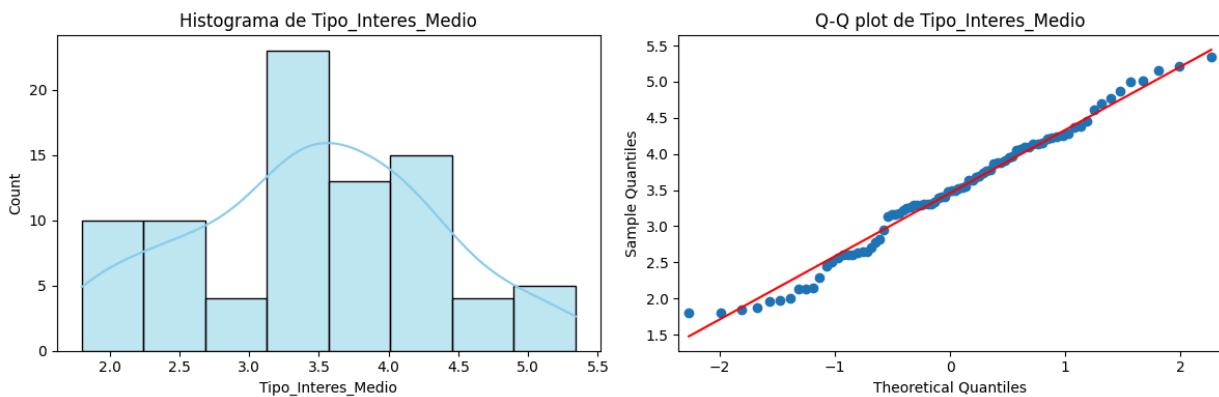
Variable: Tasa_Paro_(%)
 Shapiro-Wilk: stat = 0.9496, p = 0.0024 → ✗ No normal
 D'Agostino K²: stat = 13.7425, p = 0.0010 → ✗ No normal
 Anderson-Darling: stat = 1.0592
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✗ No normal
 ↴ 5.0%: CV=0.754 → ✗ No normal
 ↴ 2.5%: CV=0.879 → ✗ No normal
 ↴ 1.0%: CV=1.046 → ✗ No normal



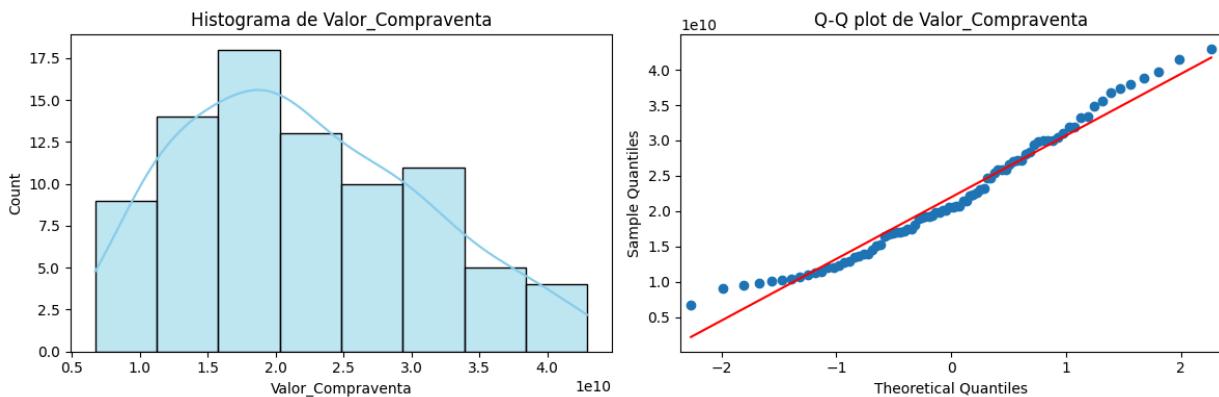
Variable: Ti_Credito
 Shapiro-Wilk: stat = 0.9276, p = 0.0002 → ✗ No normal
 D'Agostino K²: stat = 9.2012, p = 0.0100 → ✗ No normal
 Anderson-Darling: stat = 1.6215
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✗ No normal
 ↴ 5.0%: CV=0.754 → ✗ No normal
 ↴ 2.5%: CV=0.879 → ✗ No normal
 ↴ 1.0%: CV=1.046 → ✗ No normal



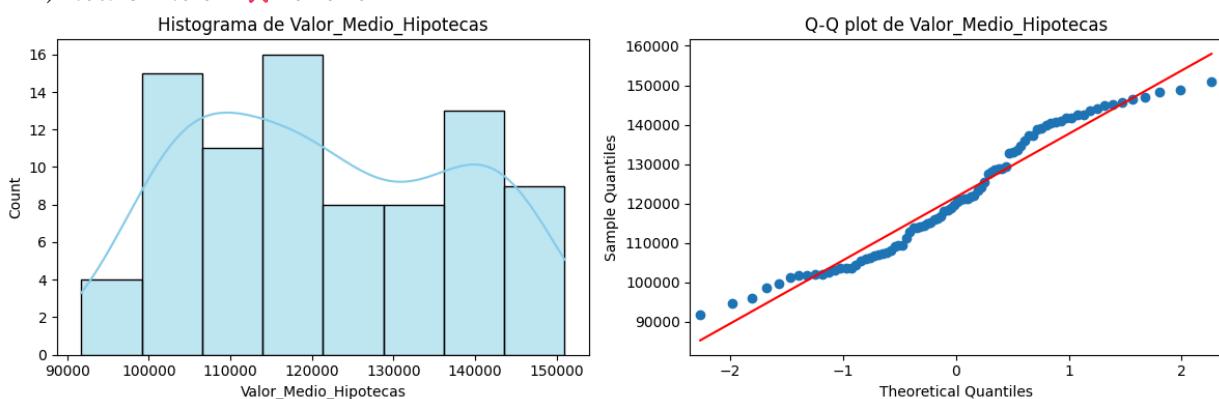
Variable: Tipo_Interes_Medio
 Shapiro-Wilk: stat = 0.9793, p = 0.1956 → ✓ Normal
 D'Agostino K²: stat = 1.6279, p = 0.4431 → ✓ Normal
 Anderson-Darling: stat = 0.3950
 ↴ 15.0%: CV=0.552 → ✓ Normal
 ↴ 10.0%: CV=0.628 → ✓ Normal
 ↴ 5.0%: CV=0.754 → ✓ Normal
 ↴ 2.5%: CV=0.879 → ✓ Normal
 ↴ 1.0%: CV=1.046 → ✓ Normal



Variable: Valor_Compraventa
 Shapiro-Wilk: stat = 0.9652, p = 0.0226 → ✗ No normal
 D'Agostino K²: stat = 4.8899, p = 0.0867 → ✓ Normal
 Anderson-Darling: stat = 0.7692
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✗ No normal
 ↴ 5.0%: CV=0.754 → ✗ No normal
 ↴ 2.5%: CV=0.879 → ✓ Normal
 ↴ 1.0%: CV=1.046 → ✓ Normal



Variable: Valor_Medio_Hipotecas
 Shapiro-Wilk: stat = 0.9479, p = 0.0019 → ✗ No normal
 D'Agostino K²: stat = 28.9625, p = 0.0000 → ✗ No normal
 Anderson-Darling: stat = 1.4044
 ↴ 15.0%: CV=0.552 → ✗ No normal
 ↴ 10.0%: CV=0.628 → ✗ No normal
 ↴ 5.0%: CV=0.754 → ✗ No normal
 ↴ 2.5%: CV=0.879 → ✗ No normal
 ↴ 1.0%: CV=1.046 → ✗ No normal



A continuación, teniendo en cuenta los resultados anteriores vamos a realizar una clasificación de todas nuestras variables (a excepción de la Fecha) en tres grupos: variables normales, aproximadamente normales y no normales. Los criterios son:

- **Variables normales:**

Shapiro-Wilk (si aplica) y D'Agostino con $p \geq 0.05$

y Anderson-Darling no supera el nivel crítico del 5%

- **Variables aproximadamente normales: criterios más laxos para considerar normalidad**

Aprueba al menos un test de Shapiro, D'Agostino o Anderson(10% o 15%)

O solo falla ligeramente en Shapiro o D'Agostino ($0.01 \leq p < 0.05$)

O Anderson falla en 5% pero pasa 10% o 15%

- **Variables no normales:**

Falla más de 1 test o Anderson supera todos los niveles críticos

```
In [10]: from scipy import stats

def clasificar_normalidad(df, excluir=['Fecha']):
    normales = []
    aprox_normales = []
    no_normales = []

    cols = df.select_dtypes(include=['float64', 'int64']).columns.difference(excluir)

    for col in cols:
        data = df[col].dropna()
        n = len(data)

        # --- Shapiro ---
        if n < 5000:
            stat_sw, p_sw = stats.shapiro(data)
        else:
            p_sw = None # No applicable

        # --- D'Agostino ---
        stat_dag, p_dag = stats.normaltest(data)

        # --- Anderson ---
        ad_result = stats.anderson(data, dist='norm')
        ad_stat = ad_result.statistic
        ad_levels = {float(level): cv for level, cv in zip(ad_result.significance_level, ad_result.critical_values)}

        # Comprobación Anderson en niveles 15% y 10%
        anderson_15 = ad_stat < ad_levels.get(15.0, float('inf'))
        anderson_10 = ad_stat < ad_levels.get(10.0, float('inf'))

        # Variables normales: todos pasan (Shapiro  $p \geq 0.05$ , D'Agostino  $p \geq 0.05$  y Anderson 5%)
        pasa_shapiro = (p_sw is None) or (p_sw >= 0.05)
        pasa_dagostino = p_dag >= 0.05
        pasa_anderson_5 = ad_stat < ad_levels.get(5.0, float('inf'))

        if pasa_shapiro and pasa_dagostino and pasa_anderson_5:
            normales.append(col)
        else:
            # Variables aproximadamente normales:
            # - Aprueba al menos uno de Shapiro, D'Agostino, Anderson(10% o 15%)
            # - O solo falla ligeramente en Shapiro o D'Agostino ( $0.01 \leq p < 0.05$ )
            # - O Anderson falla en 5% pero pasa 10% o 15%
            aprox_normal = False

            if (p_sw is not None and p_sw >= 0.05) or (p_dag >= 0.05) or anderson_10 or anderson_15:
                aprox_normal = True
            elif (p_sw is not None and 0.01 <= p_sw < 0.05) or (0.01 <= p_dag < 0.05):
                aprox_normal = True
            elif (not pasa_anderson_5) and (anderson_10 or anderson_15):
                aprox_normal = True

            if aprox_normal:
                aprox_normales.append(col)
            else:
                no_normales.append(col)

    return normales, aprox_normales, no_normales
```

```
In [11]: def imprimir_lista_titulo(titulo, lista):
    print(f"\n{'*'*40}\n{titulo} (Total: {len(lista)})\n{'*'*40}")
    if lista:
        for i, var in enumerate(lista, 1):
            print(f"{i}. {var}")
    else:
        print("Ninguna variable en esta categoría.")

# Clasificación
normales, aprox_normales, no_normales = clasificar_normalidad(df_clean, excluir=['Fecha'])
```

```

# Mostrar resultados
imprimir_lista_titulo("Variables normales", normales)
imprimir_lista_titulo("Variables aproximadamente normales", aprox_normales)
imprimir_lista_titulo("Variables no normales", no_normales)

=====
Variables normales (Total: 2)
=====
1. Renta_Disponible_Bruta
2. Tipo_Interes_Medio

=====
Variables aproximadamente normales (Total: 10)
=====
1. Afiliaciones_Ss
2. Contratos_Temporales
3. Euribor_12_Meses
4. Interes_Variable_Hipotecas
5. Numero_Compraventas
6. Pib_Trimestral_Ajustado
7. Pib_Trimestral_No_Ajustado
8. Precio_M2_Vivienda
9. Ti_Credito
10. Valor_Compraventa

=====
Variables no normales (Total: 14)
=====
1. Cantidad_De_Extranjeros
2. Confianza_Del_Consumidor
3. Contratos_Indefinidos
4. Duracion_Media_Hipoteca
5. Importe_Hipotecas
6. Interes_Fijo_Hipotecas
7. Ipc_%_Variación_Anual
8. Ipi_Variacion_Anual_Corregida
9. Ipi_Variacion_Anual_Original
10. Mro_Rate
11. Numero_Hipotecas
12. Poblacion
13. Tasa_Paro_()
14. Valor_Medio_Hipotecas

```

3. Análisis temporal

Evolución temporal y tendencia

El objetivo aquí es ver cómo se comporta la variable a lo largo del período de análisis, es decir, representarla en función del tiempo. Esto nos permite visualizar una evolución general de la misma e identificar caídas o picos más importantes, tendencias generales o patrones estacionales visibles

```

In [12]: df_clean['Fecha'] = pd.to_datetime(df_clean['Fecha'])
df_clean = df_clean.sort_values('Fecha').reset_index(drop=True)

df_clean['Fecha_ordinal'] = df_clean['Fecha'].map(pd.Timestamp.toordinal)

cols = df_clean.select_dtypes(include=['float64', 'int64']).columns.difference(['Fecha_ordinal'])

plt.figure(figsize=(14, len(cols) * 3))
for i, col in enumerate(cols, 1):
    plt.subplot(len(cols), 1, i)

    sns.lineplot(data=df_clean, x='Fecha', y=col, label='Serie temporal')

    mask = df_clean[col].notnull()
    x = df_clean.loc[mask, 'Fecha_ordinal'].values.reshape(-1, 1)
    y = df_clean.loc[mask, col].astype(float).values

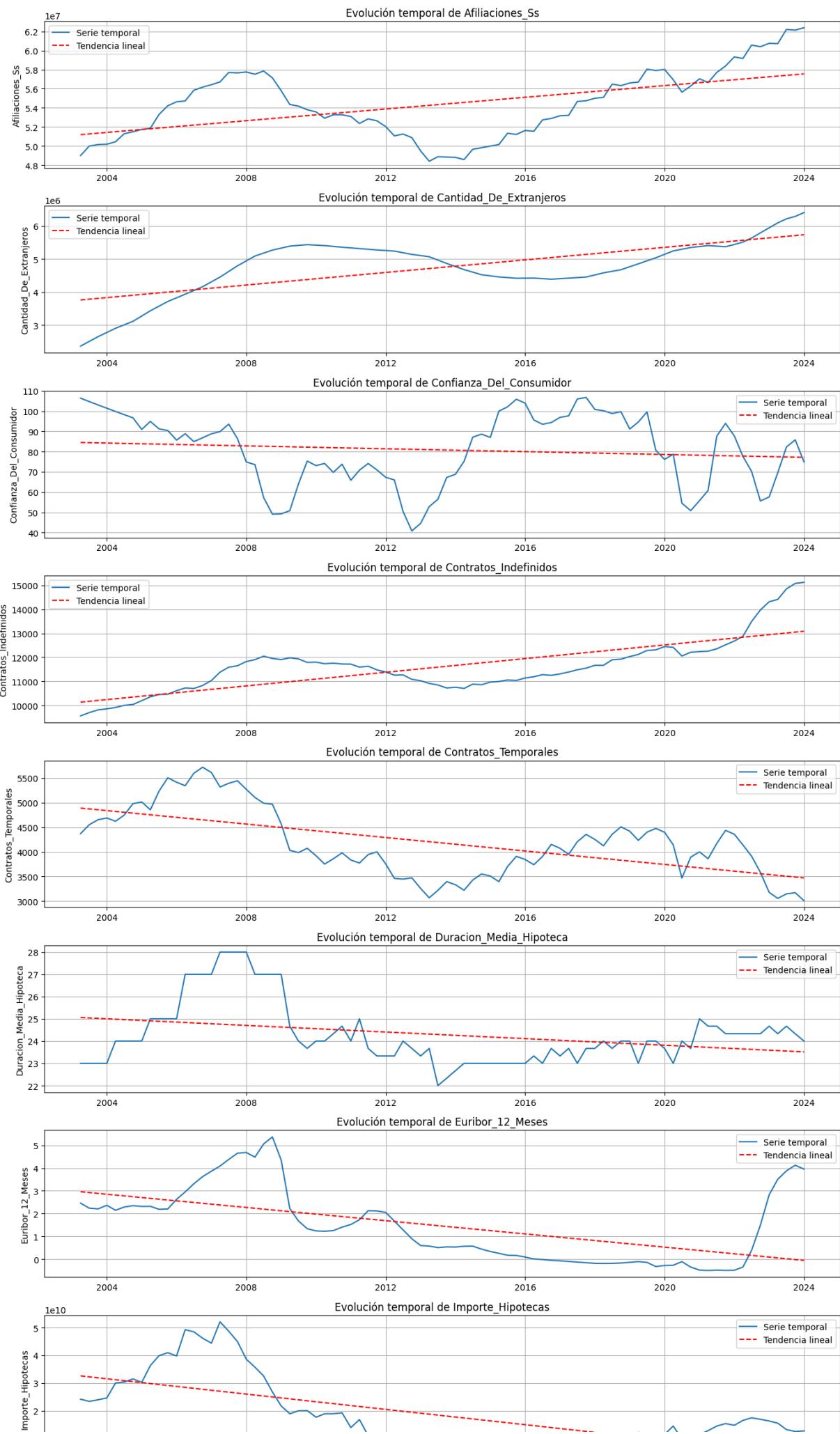
    if len(y) > 1:
        model = LinearRegression()
        model.fit(x, y)
        y_pred = model.predict(x)
        plt.plot(df_clean.loc[mask, 'Fecha'], y_pred, color='red', linestyle='--', label='Tendencia lineal')

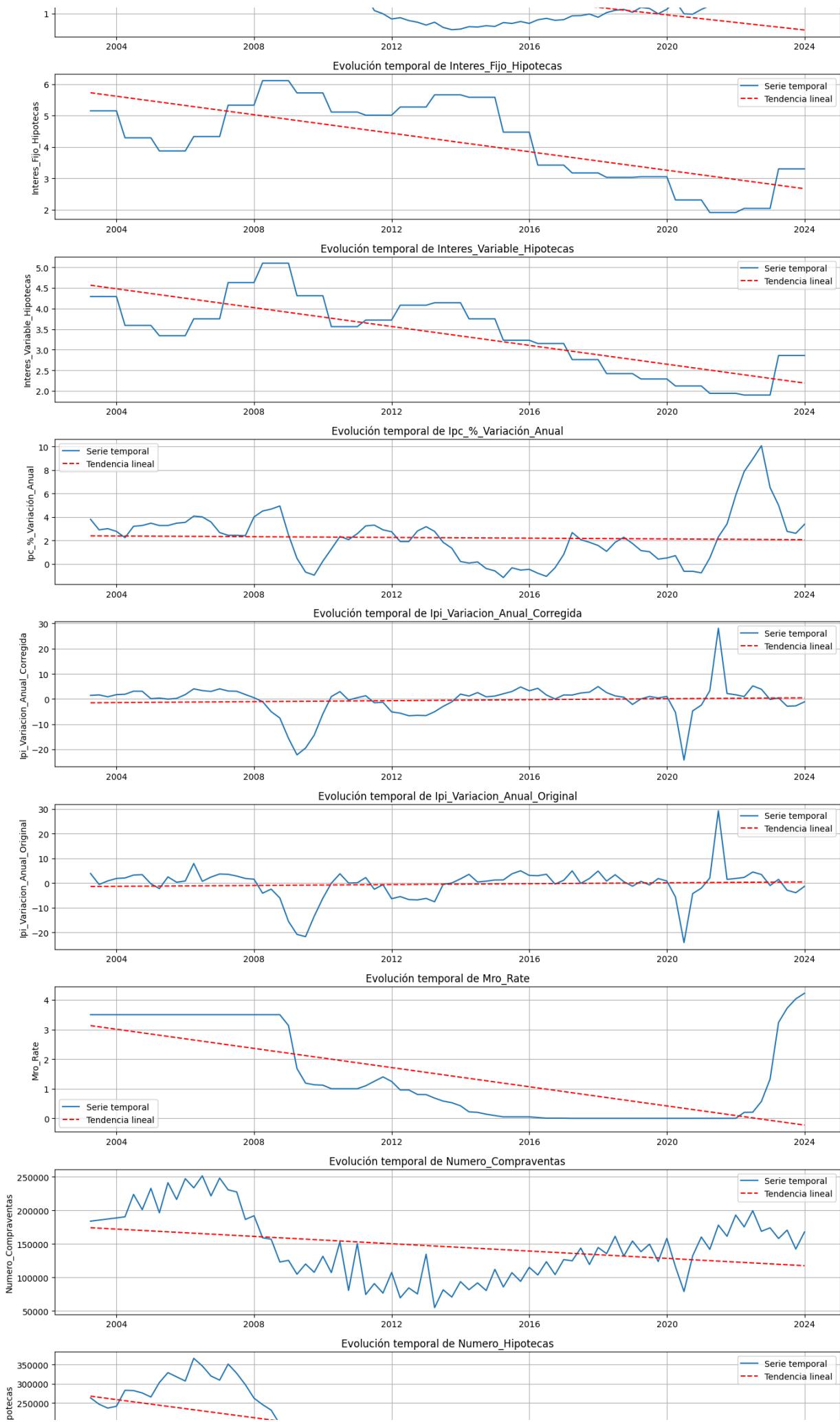
    plt.title(f"Evolución temporal de {col}")
    plt.xlabel('')

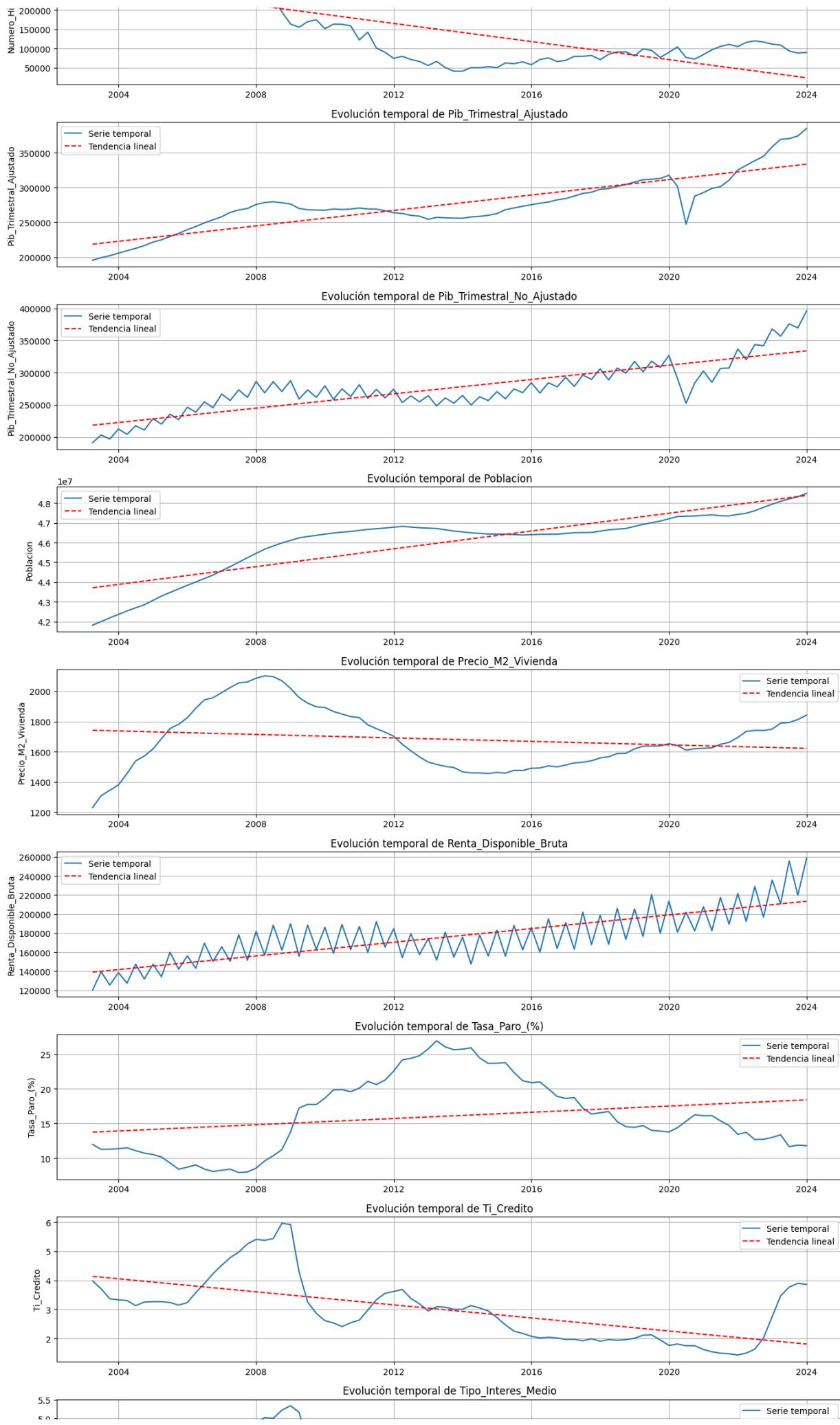
```

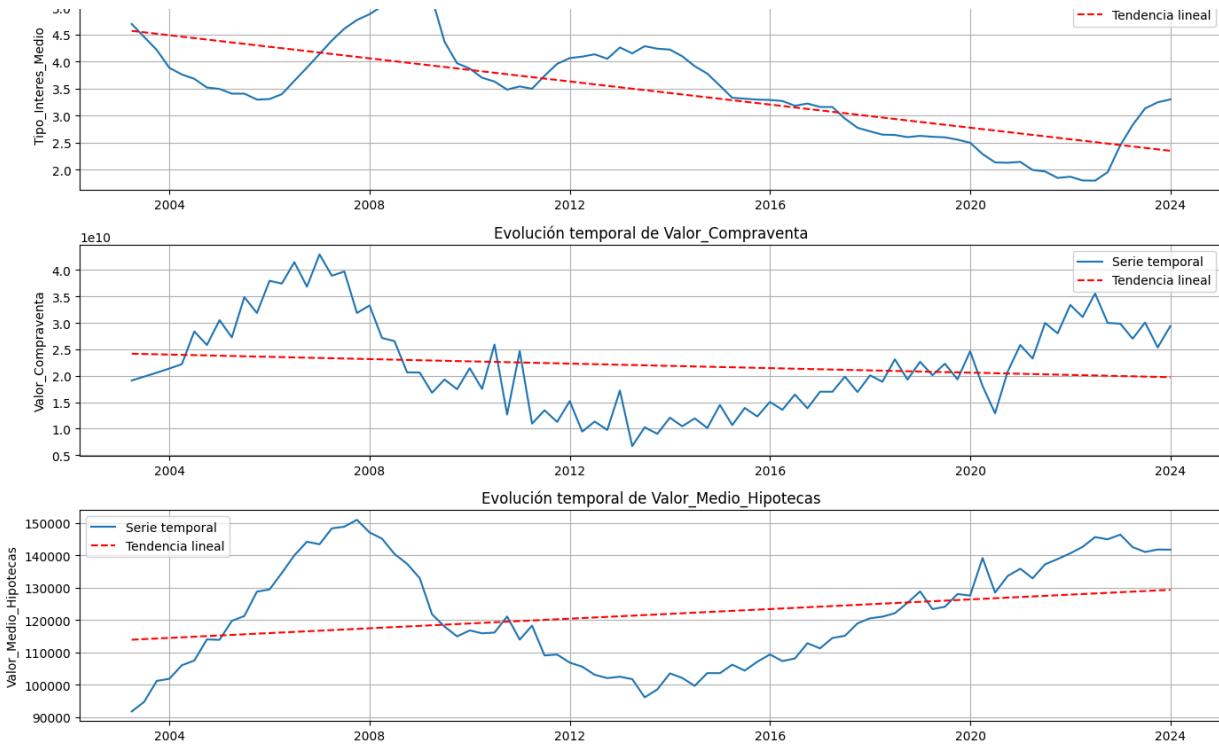
```
plt.ylabel(col)
plt.legend()
plt.grid(True)

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









Descomposición de las series temporales en tendencia, estacionalidad y residuos

A continuación, vamos a descomponer cada una de las series en tres componentes:

- **Serie observada:** es la serie original
- **Tendencia:** es la dirección general a largo plazo de la serie. Ayuda a identificar si la variable crece, decrece o se mantiene estable a largo plazo
- **Estacionalidad:** es el patrón que se repite en intervalos regulares, es decir, el efecto "cíclico" que se repite cada año, cada trimestre, etc
- **Residuo:** es lo que queda después de quitar la tendencia y la estacionalidad, es decir, la "parte aleatoria" o ruido de la serie. Ayuda a detectar anomalías o eventos inesperados. Si los residuos tienen picos grandes, puede indicar crisis económicas, cambios regulatorios o shocks puntuales

```
In [13]: for col in cols:
    print(f"\n--- Descomposición de la serie temporal: {col} ---")
    try:
        result = seasonal_decompose(df_clean[col], model='additive', period=4)

        fig, axs = plt.subplots(4, 1, figsize=(12, 10), sharex=True)

        fechas = df_clean['Fecha']
        trimestres = fechas.dt.to_period("Q").astype(str)

        axs[0].plot(fechas, result.observed)
        axs[0].set_title(f'{col} - Observada')

        axs[1].plot(fechas, result.trend)
        axs[1].set_title('Tendencia')

        axs[2].plot(fechas, result.seasonal)
        axs[2].set_title('Estacionalidad')

        axs[3].plot(fechas, result.resid)
        axs[3].set_title('Residuo')

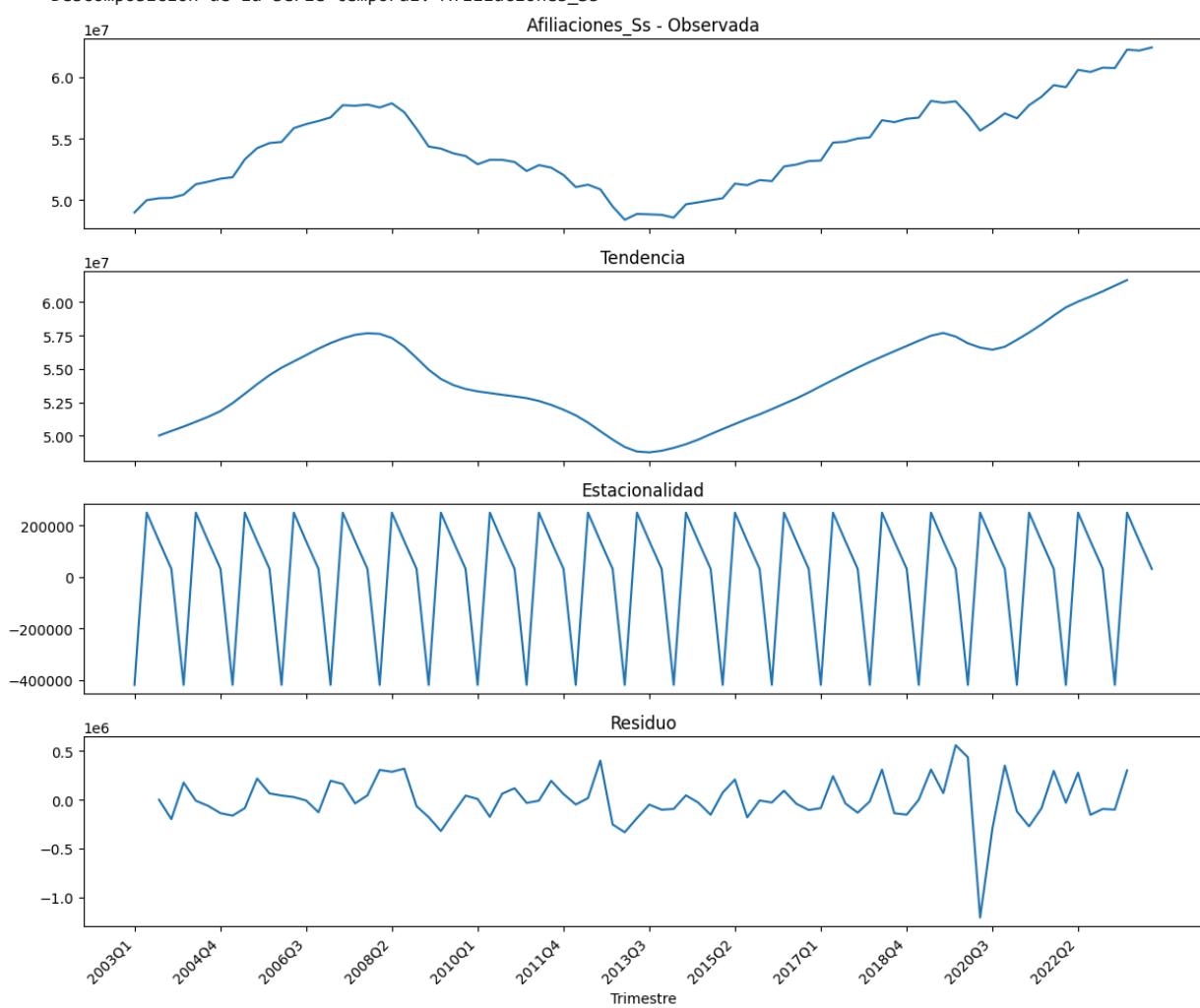
        step = max(1, len(fechas) // 12)
        axs[3].set_xticks(fechas[::step])
        axs[3].set_xticklabels(trimestres[::step], rotation=45, ha='right')

        plt.xlabel('Trimestre')
        plt.tight_layout()
```

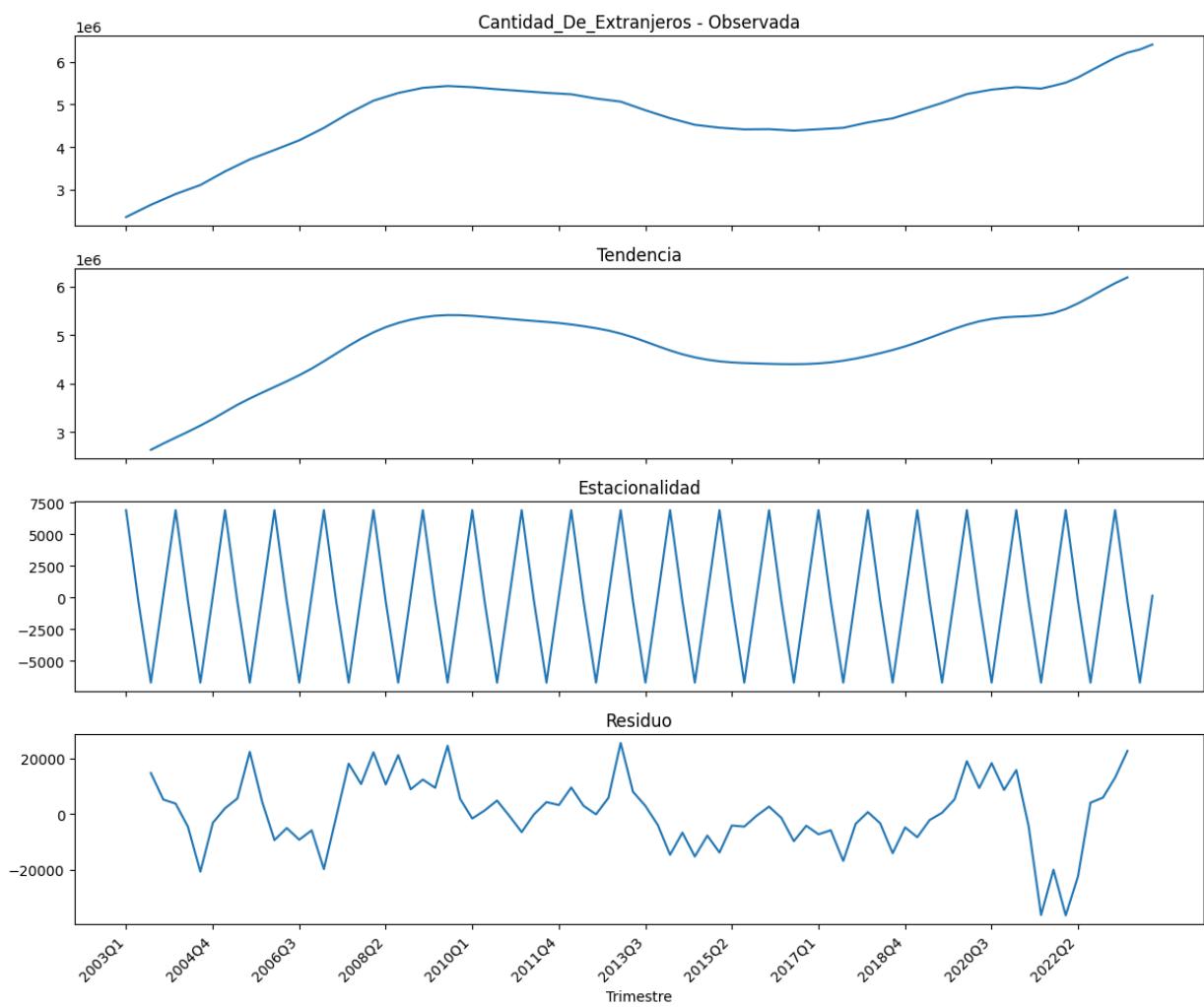
```
plt.show()

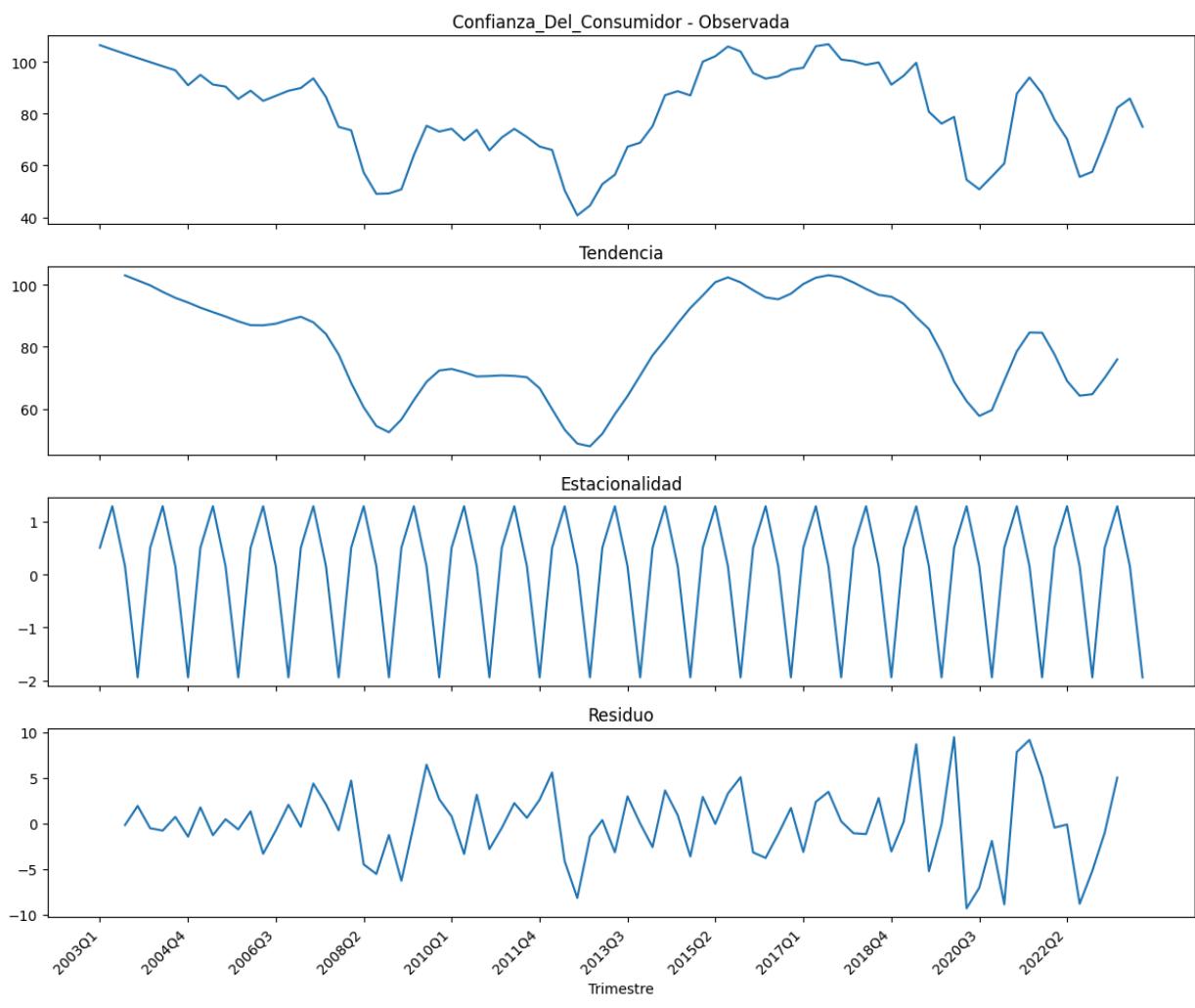
except Exception as e:
    print(f"No se pudo descomponer {col}: {e}")

--- Descomposición de la serie temporal: Afiliaciones_Ss ---
```

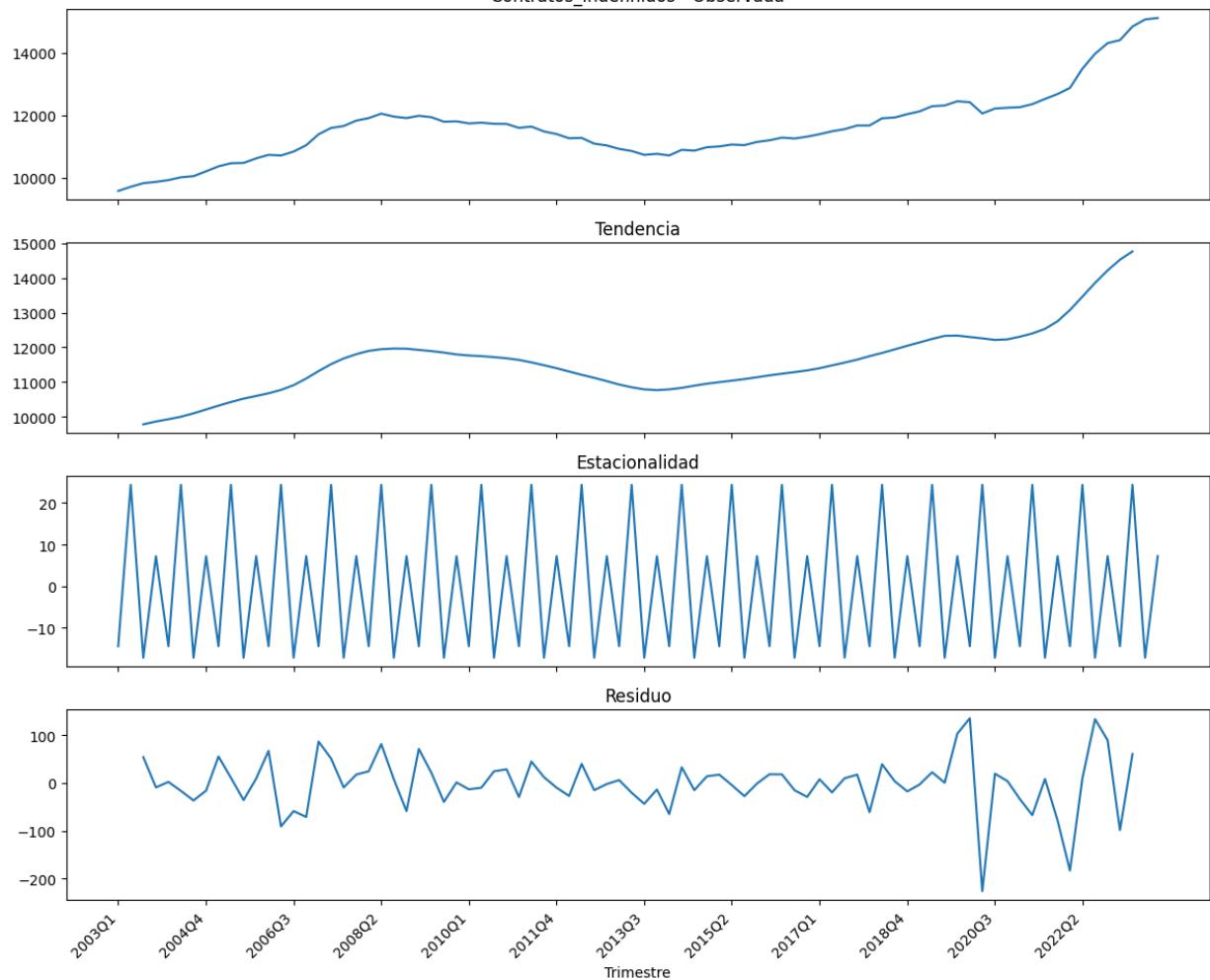


```
--- Descomposición de la serie temporal: Cantidad_De_Extranjeros ---
```

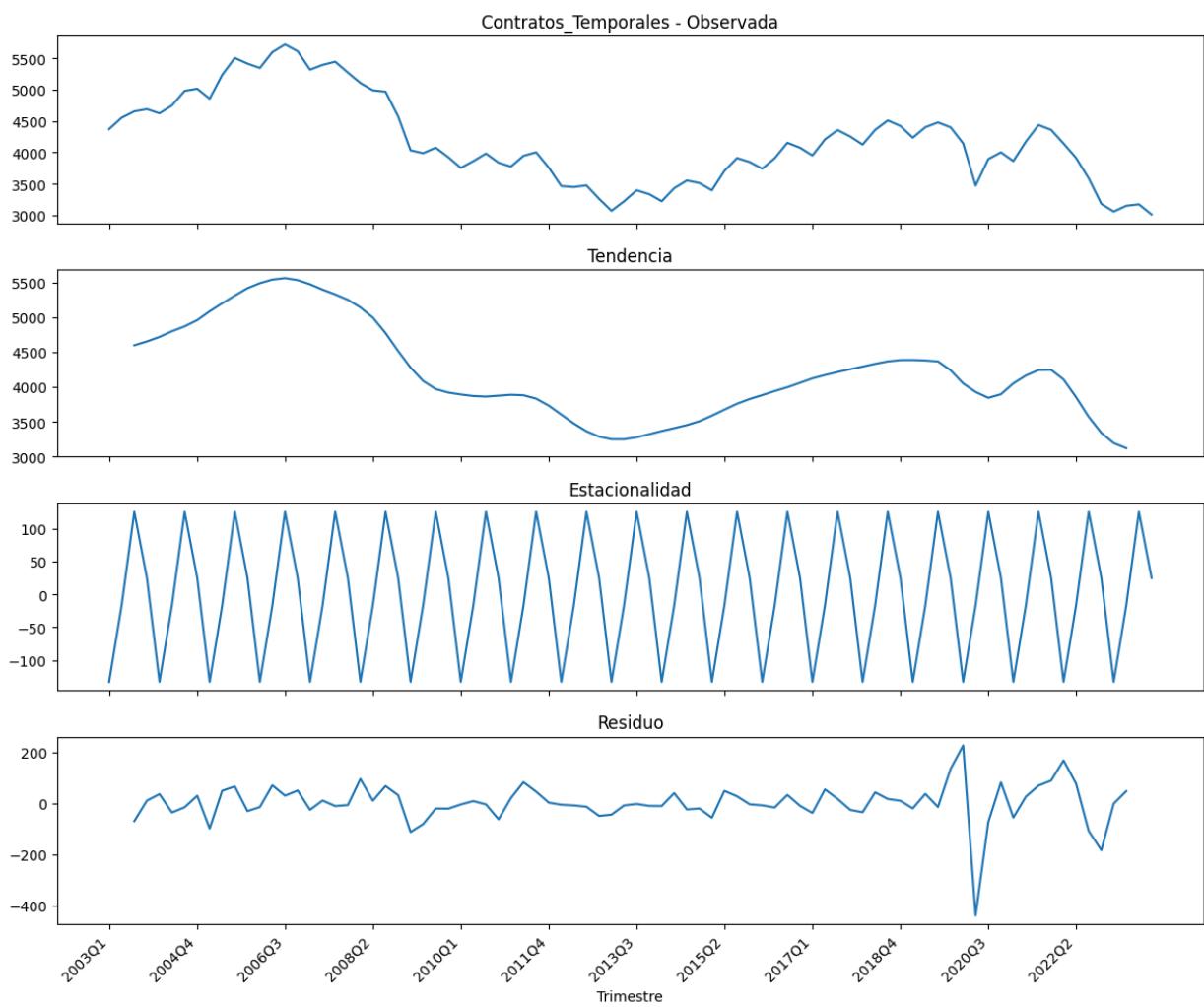


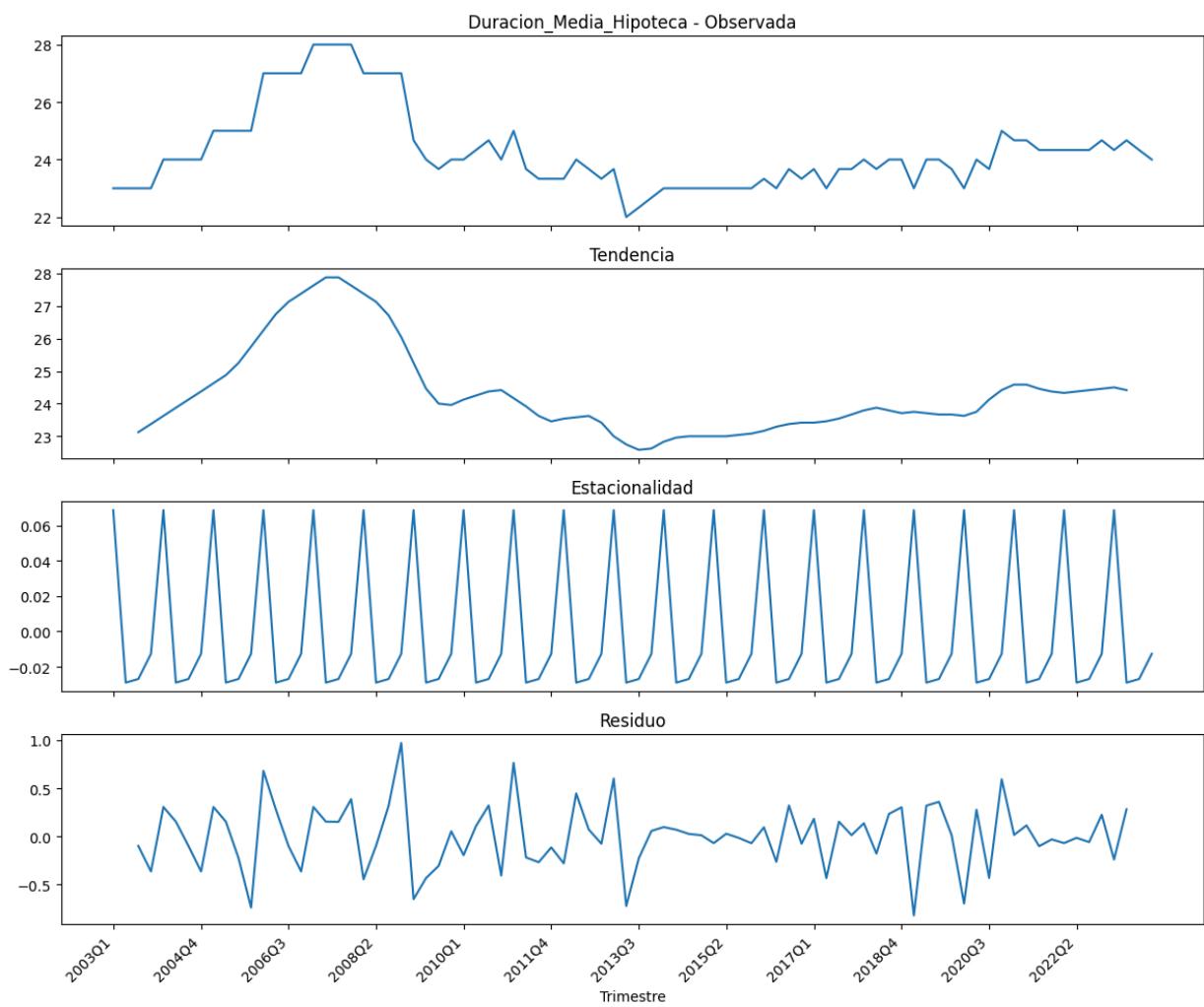


Contratos_Indefinidos - Observada

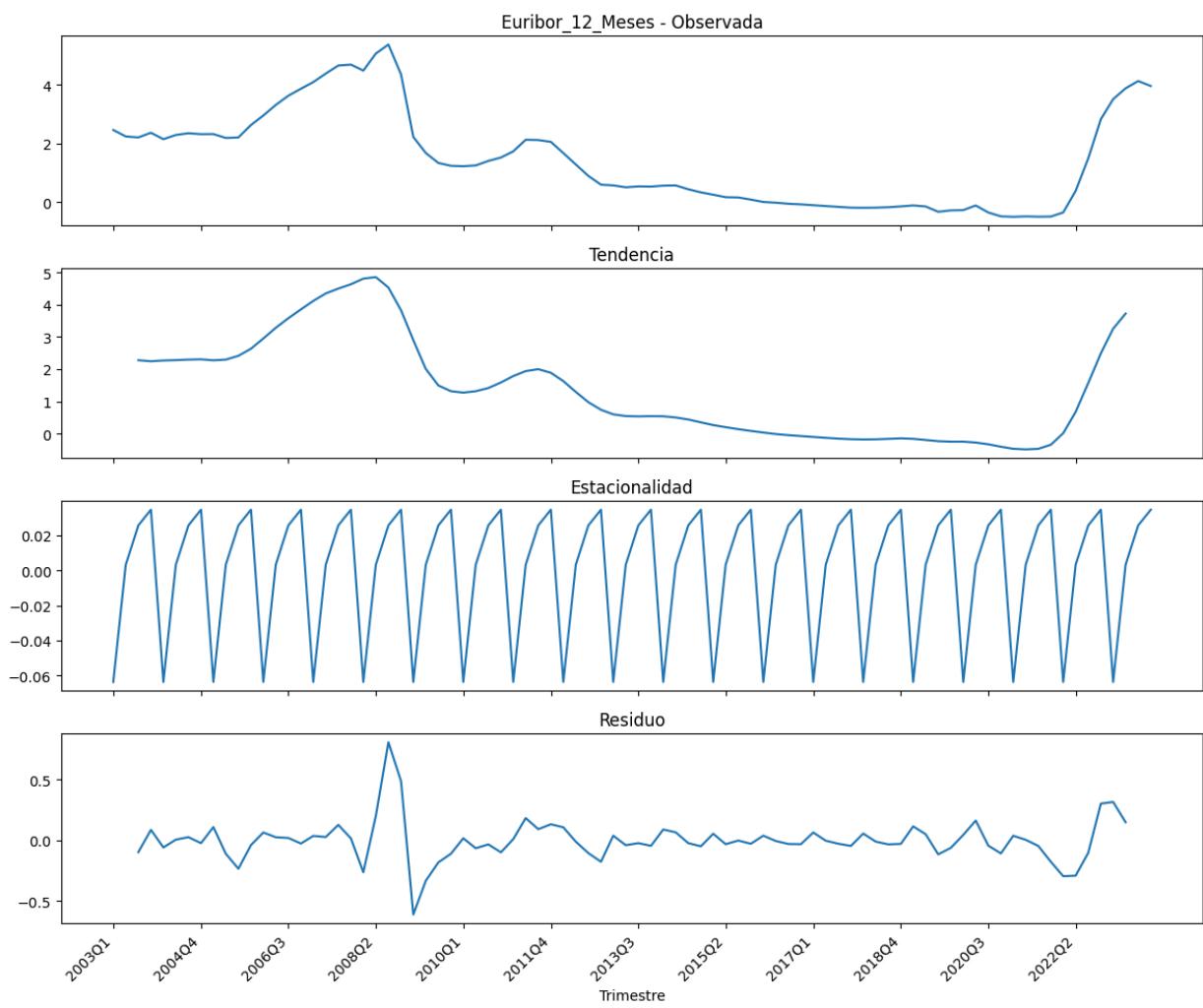


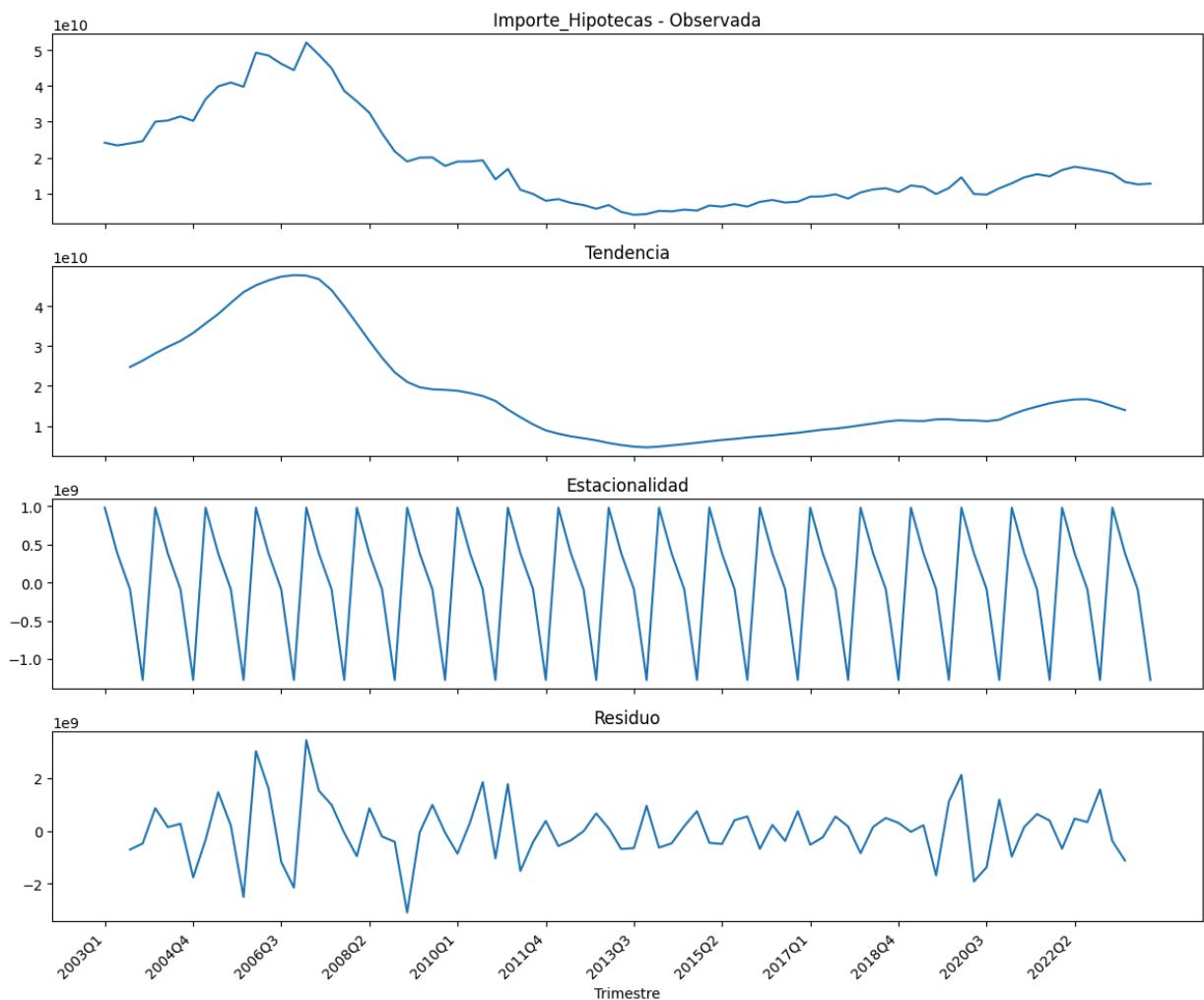
--- Descomposición de la serie temporal: Contratos_Temporales ---



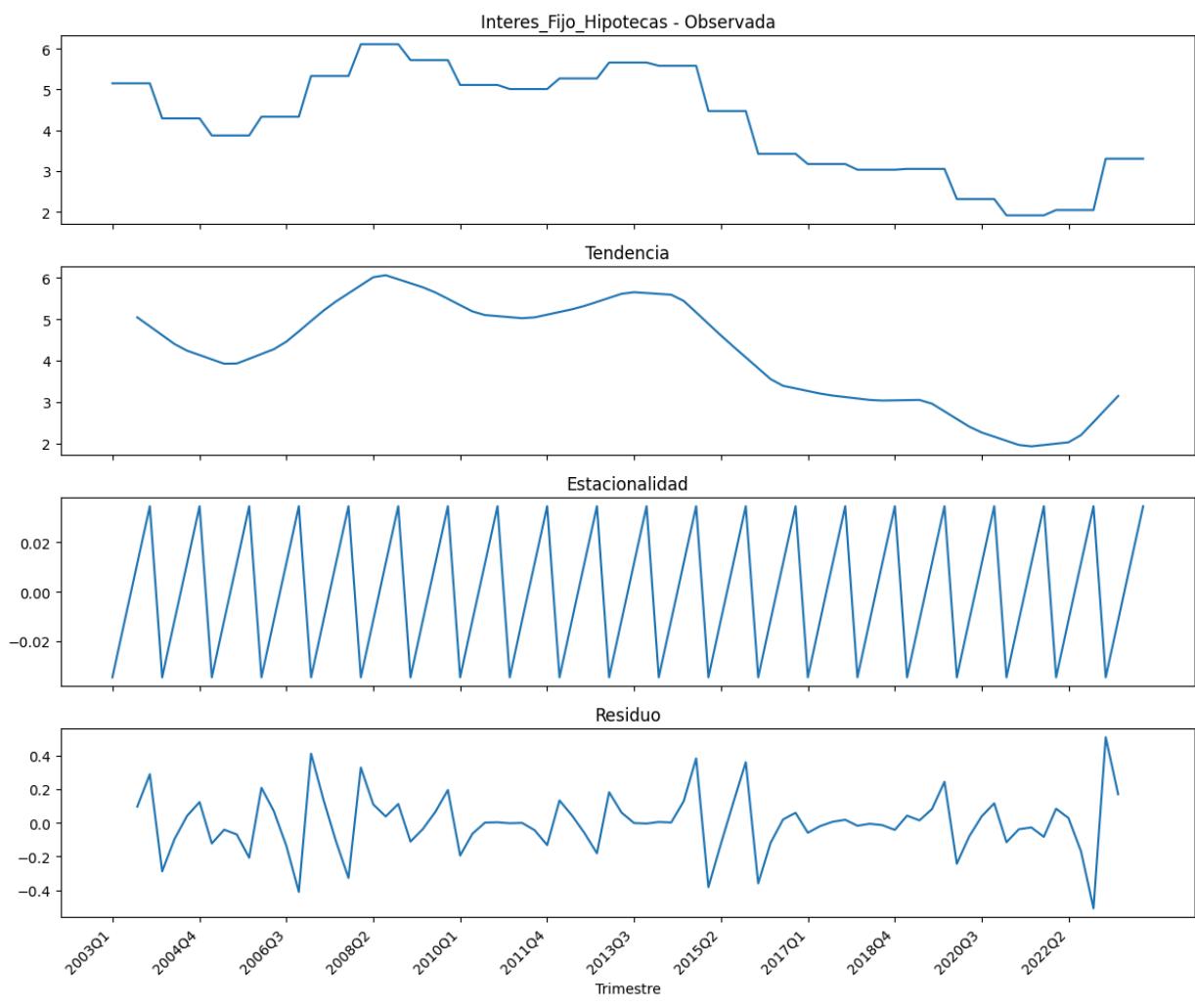


--- Descomposición de la serie temporal: Euribor_12_Meses ---

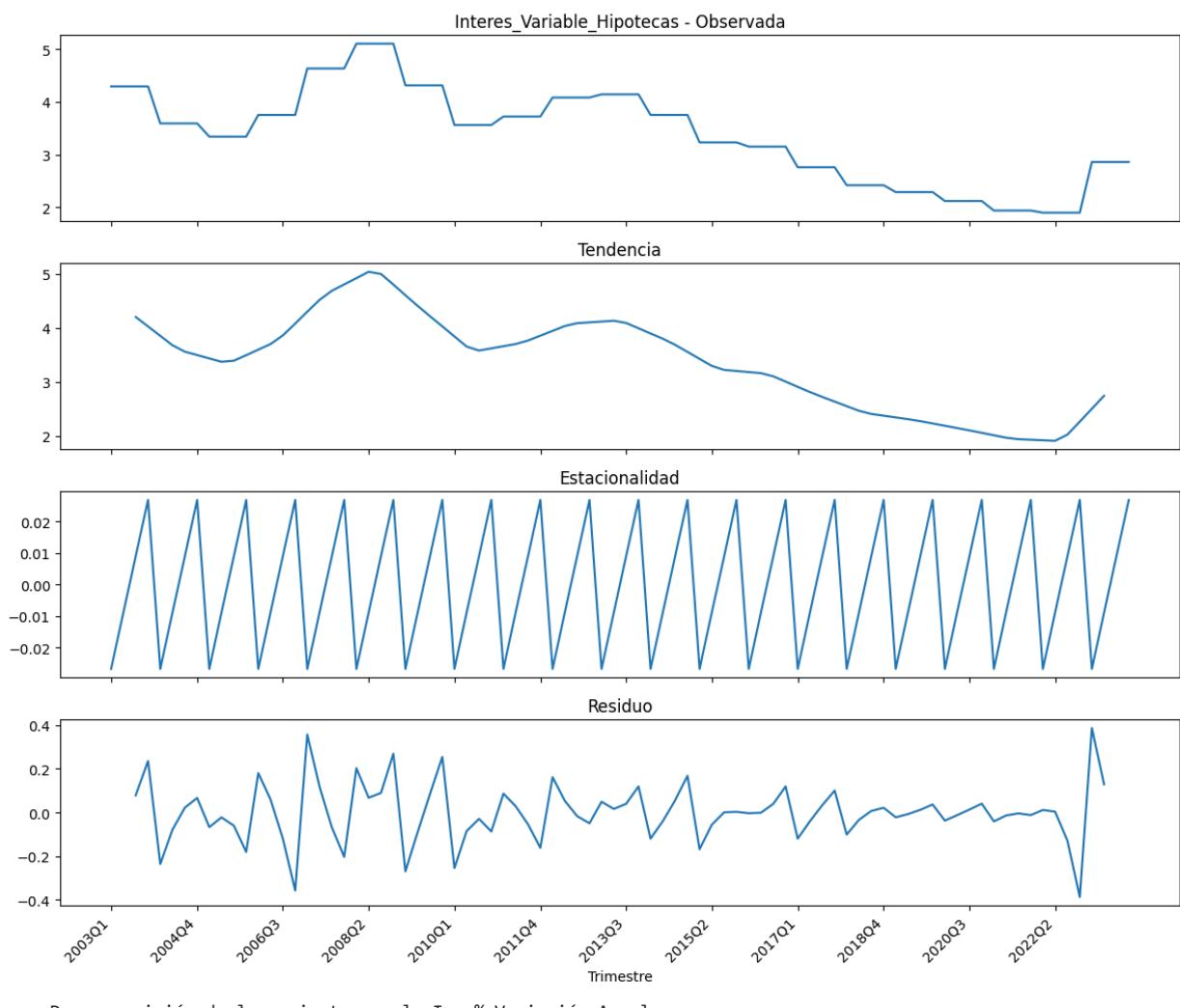




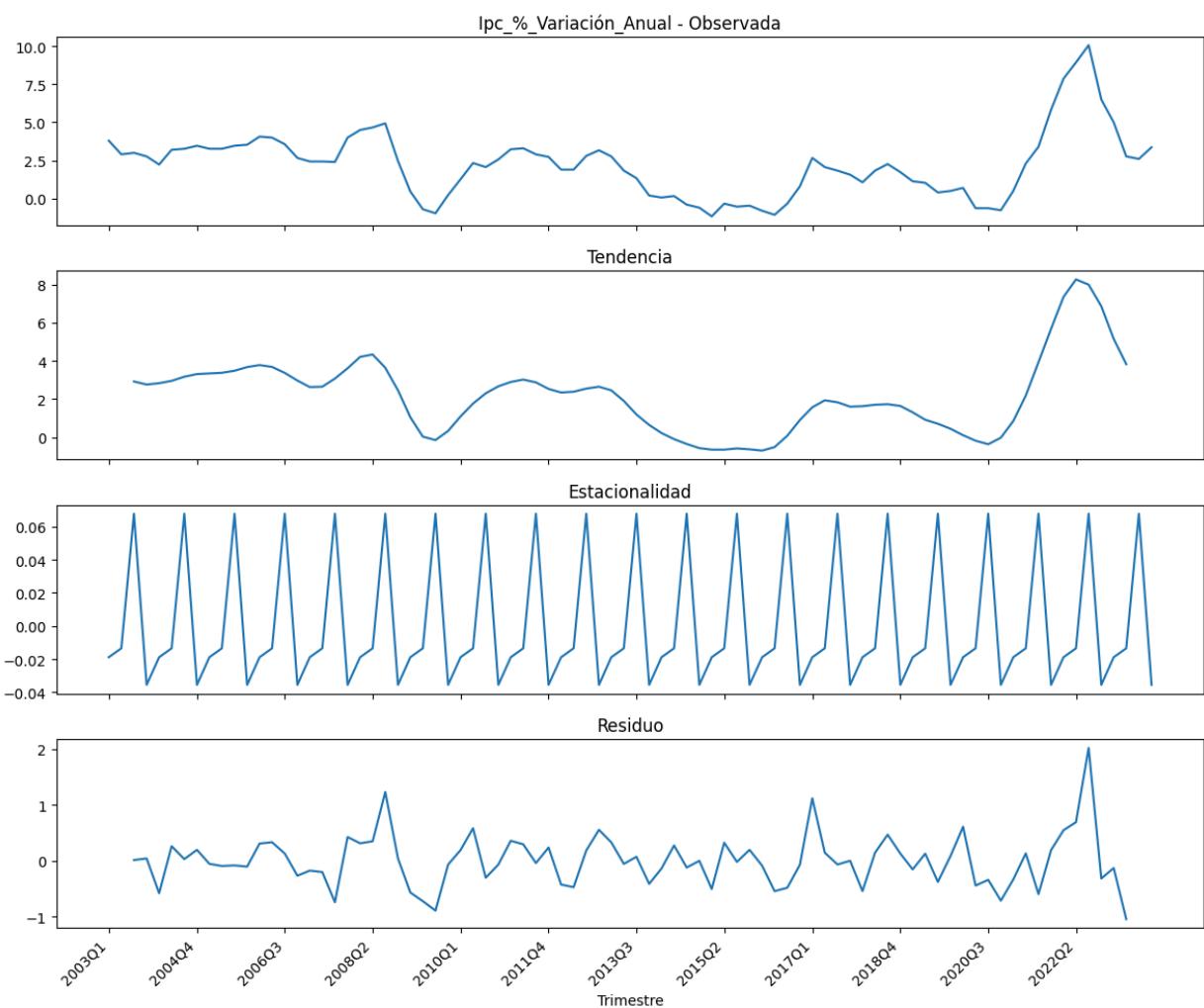
--- Descomposición de la serie temporal: Interes_Fijo_Hipotecas ---

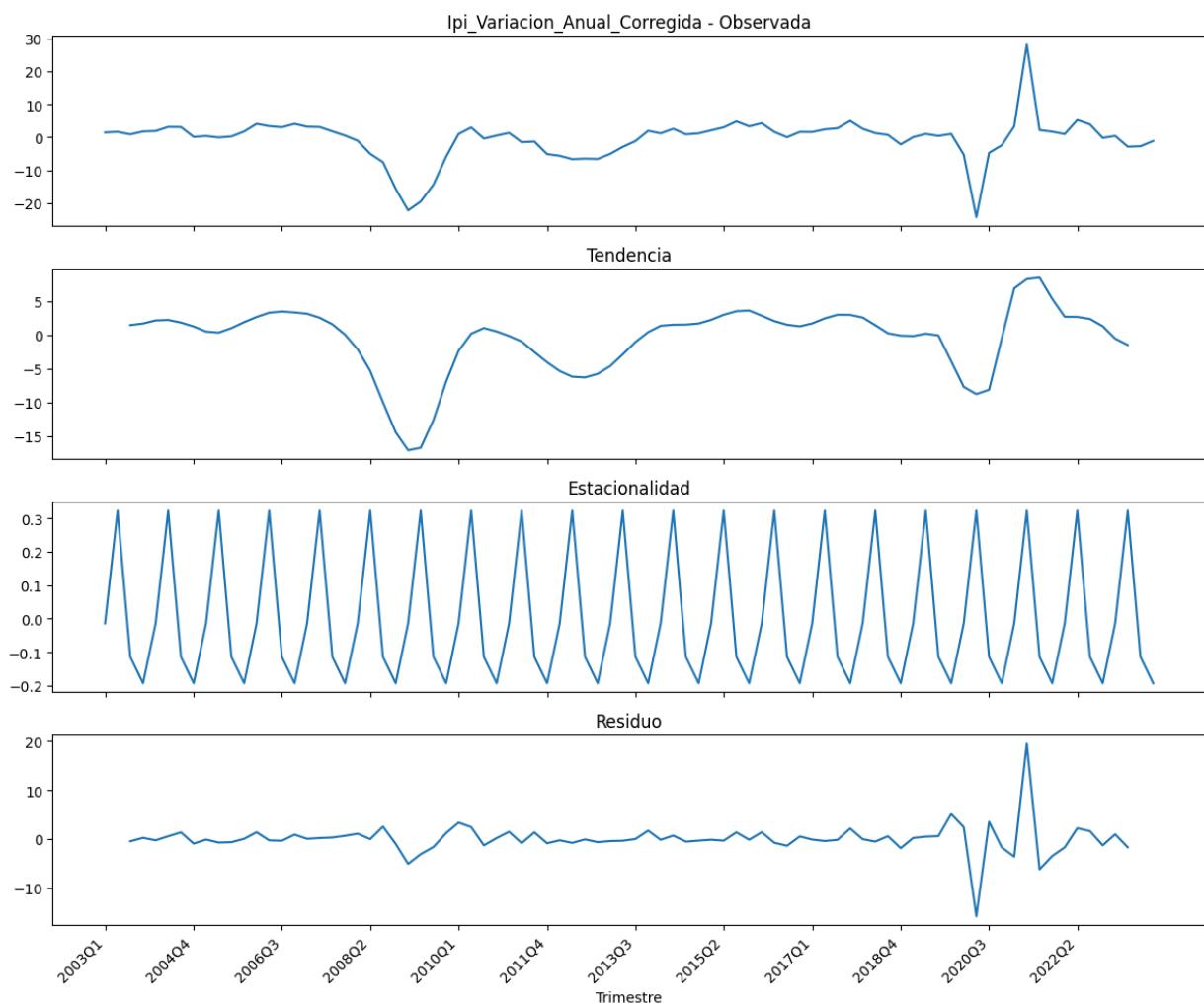


--- Descomposición de la serie temporal: Interes_Variable_Hipotecas ---



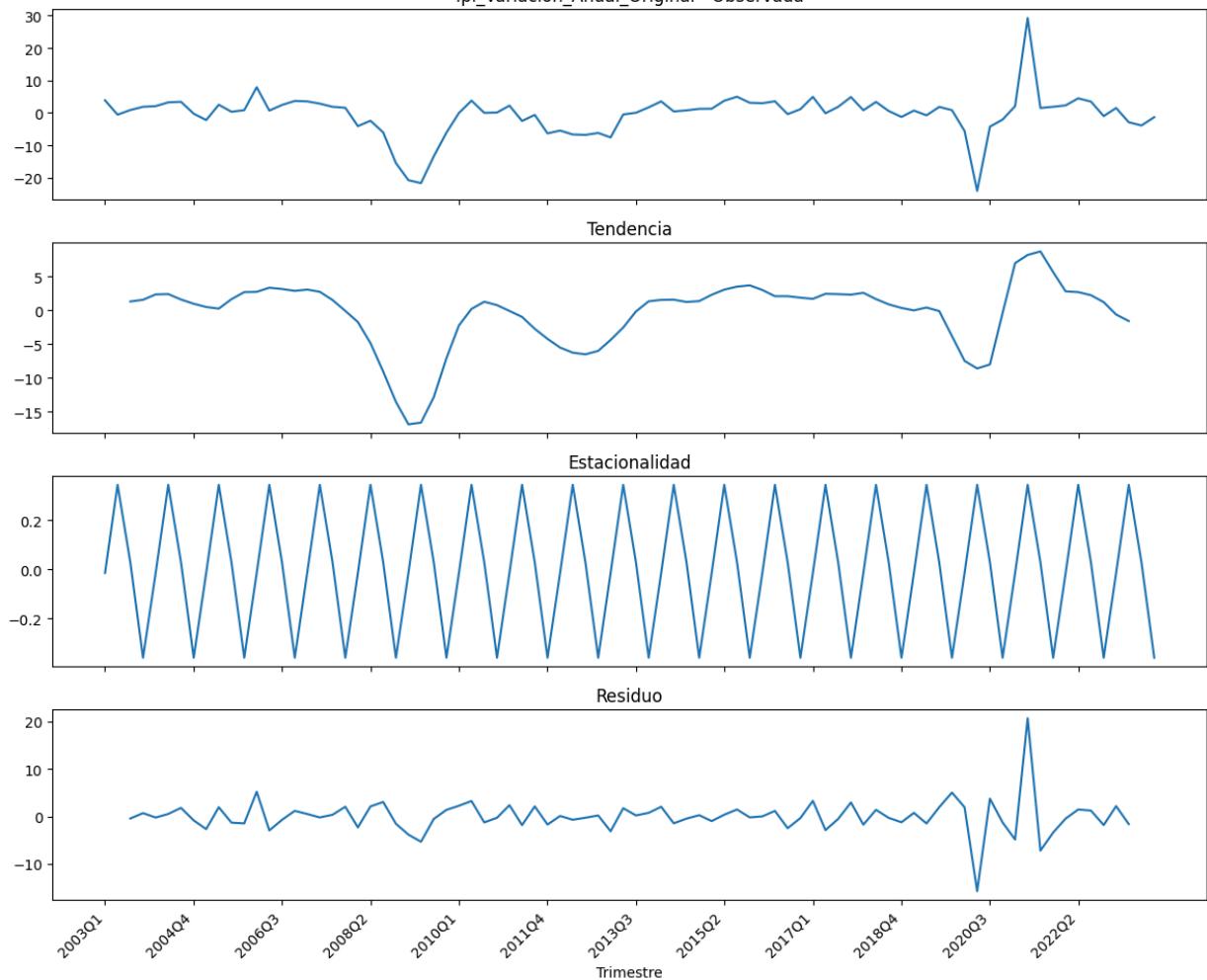
--- Descomposición de la serie temporal: Ipc_%_Variación_Annual ---



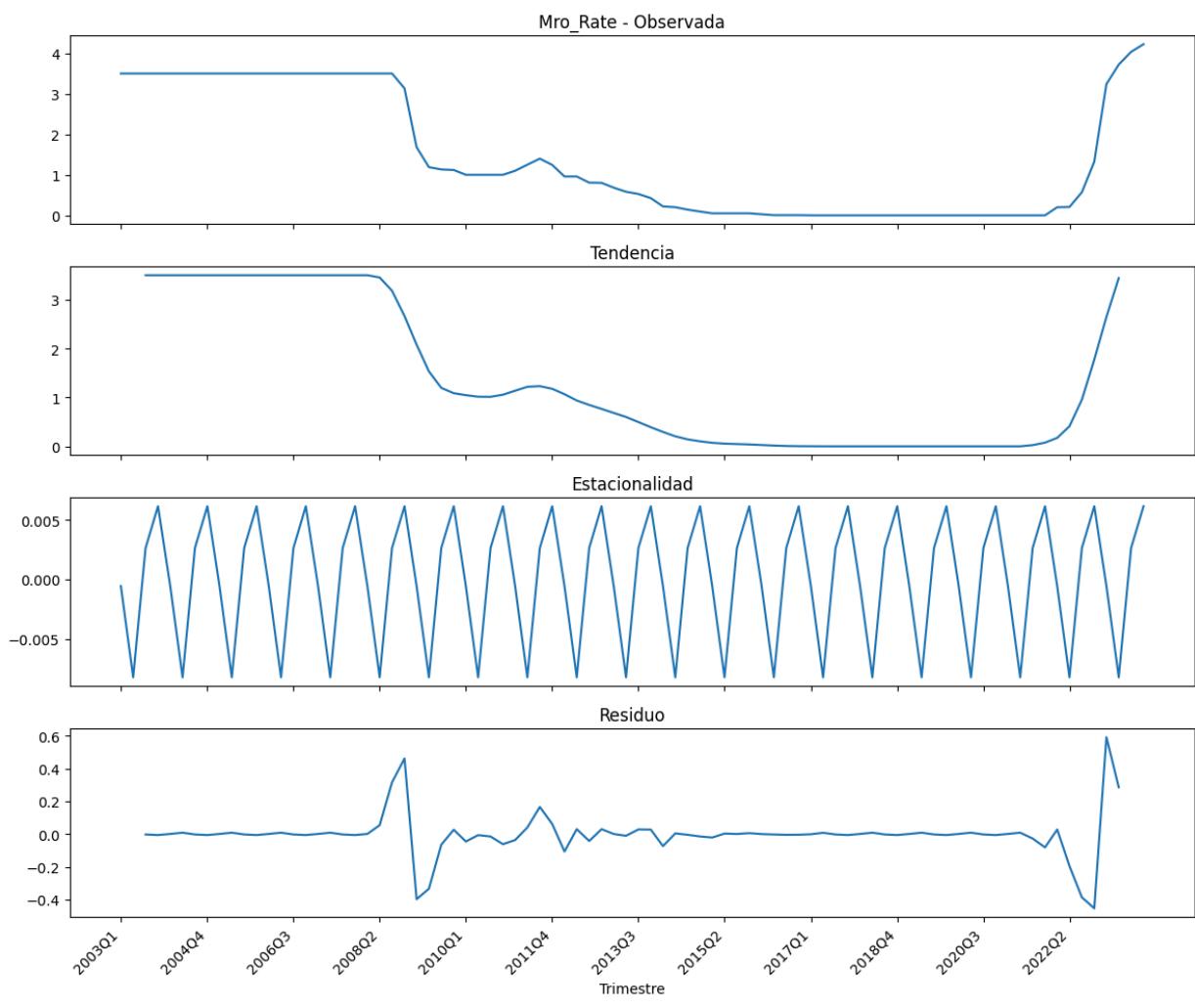


--- Descomposición de la serie temporal: Ipi_Variacion_Anual_Original ---

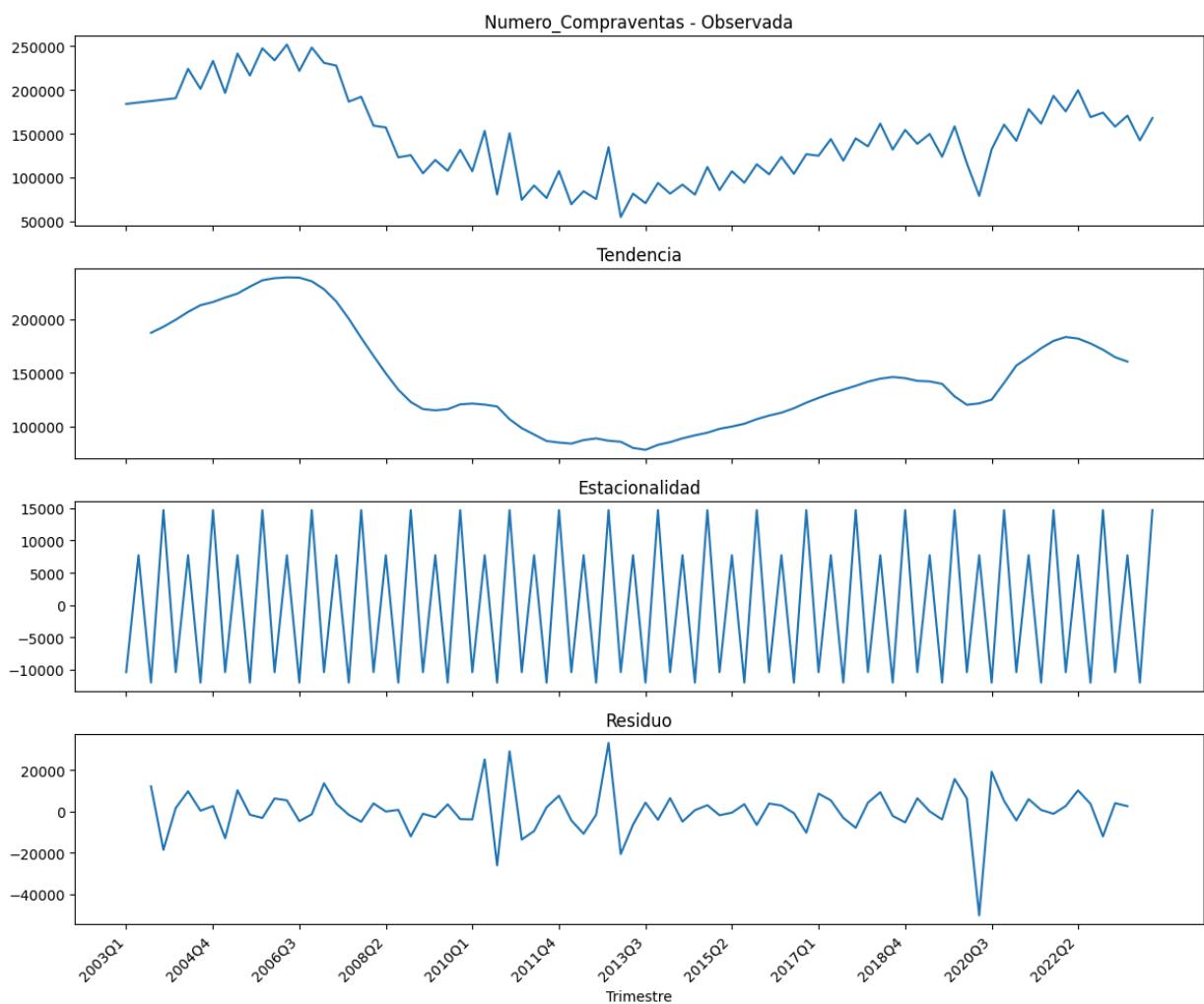
Ipi_Variacion_Anual_Original - Observada

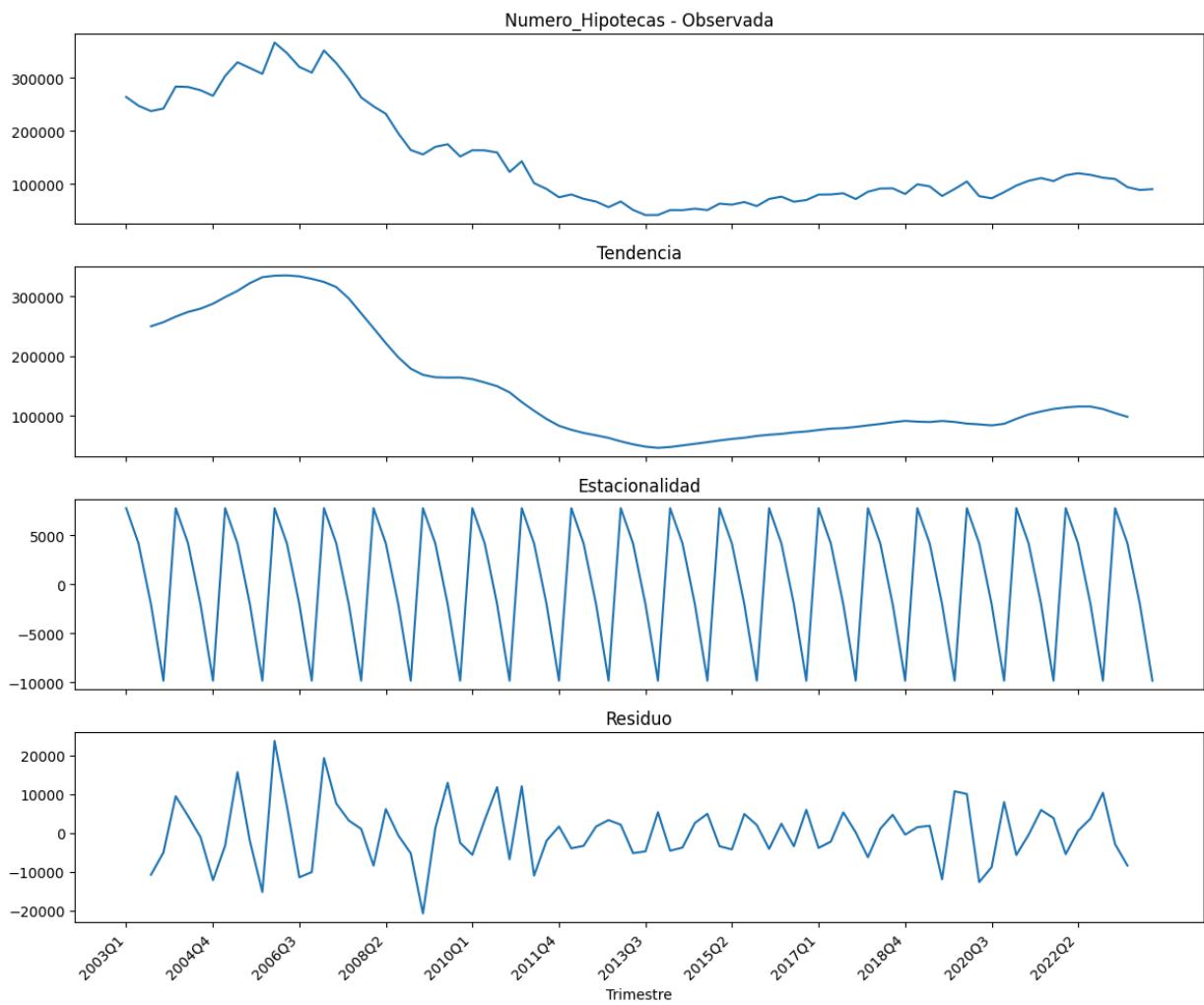


--- Descomposición de la serie temporal: Mro_Rate ---

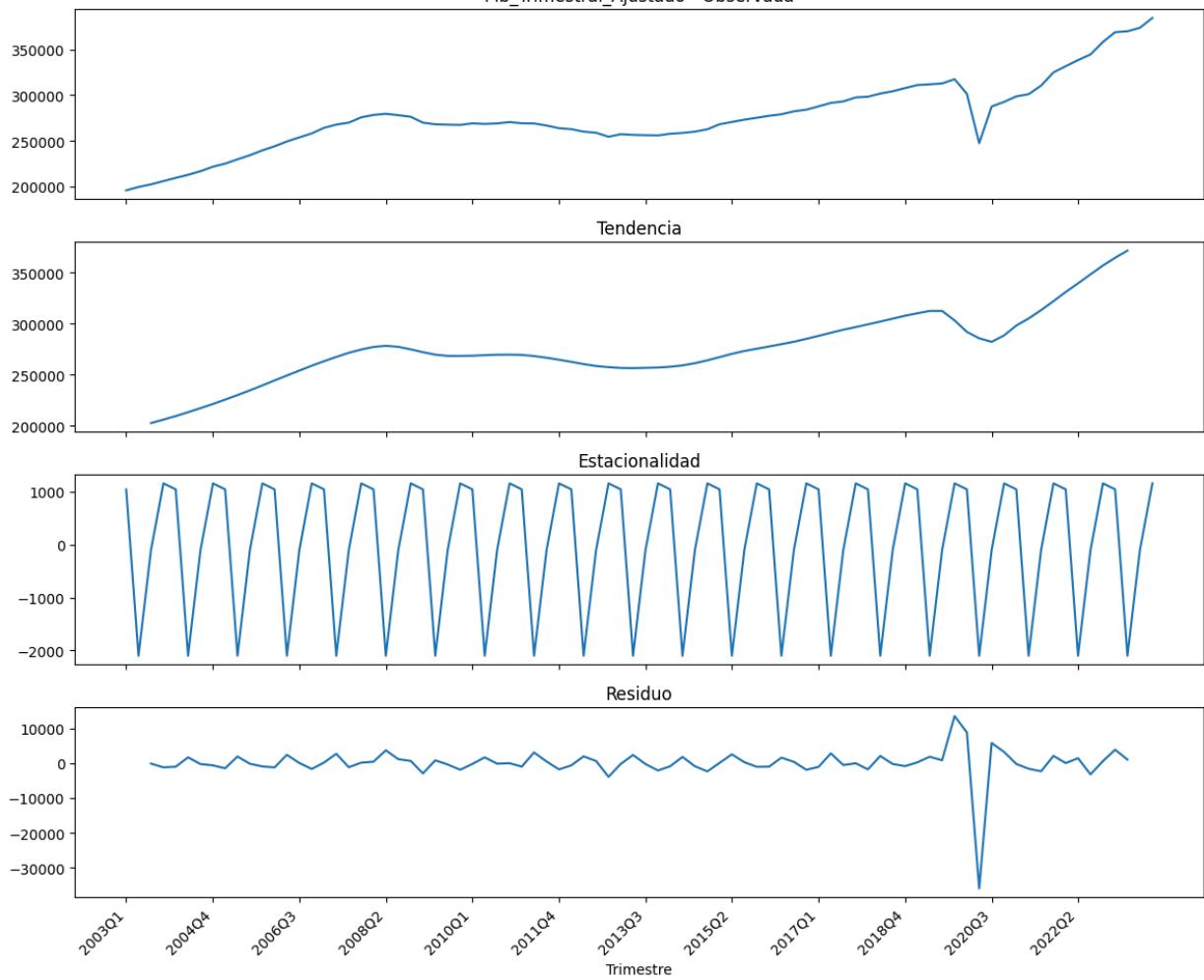


--- Descomposición de la serie temporal: Numero_Compraventas ---

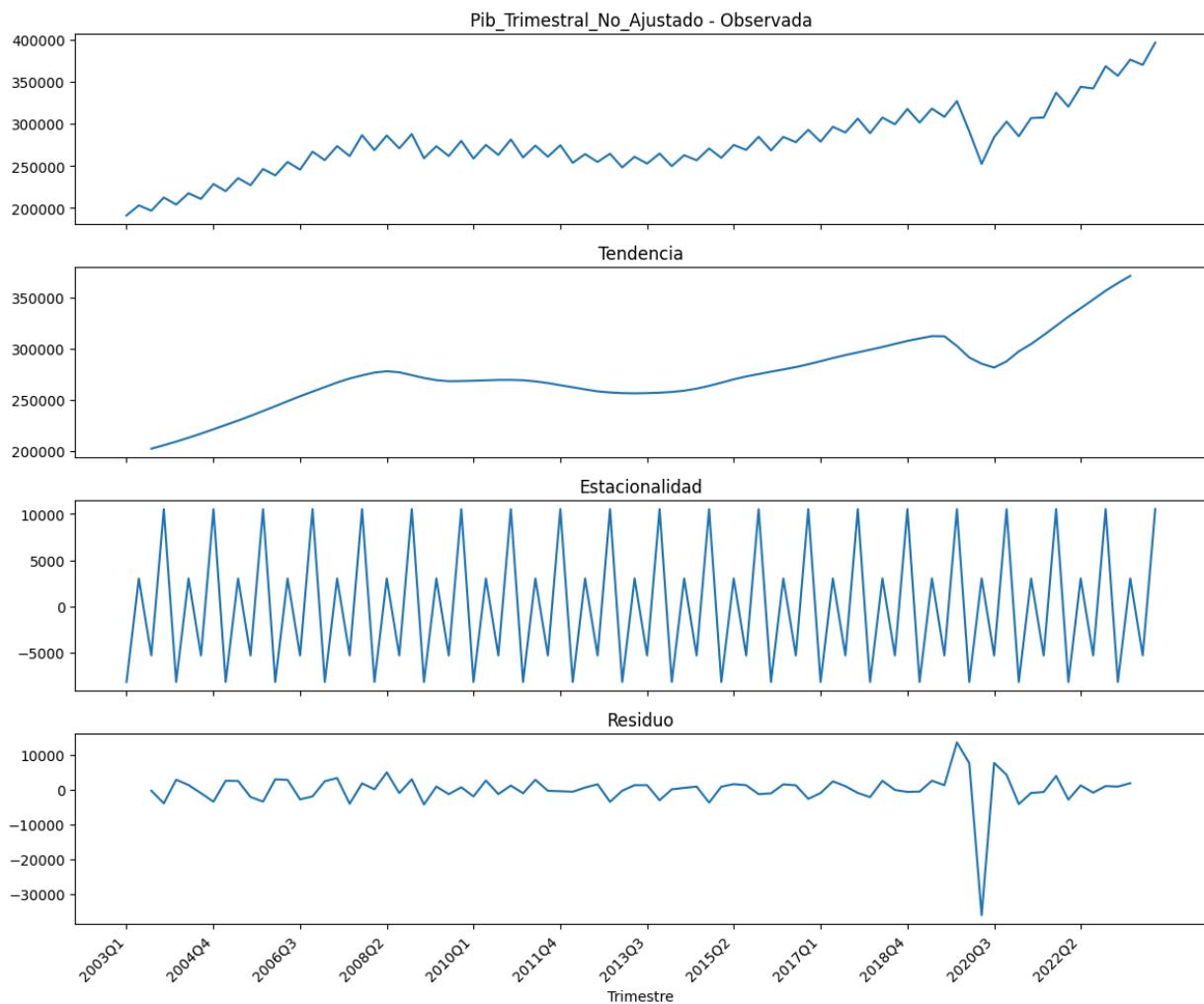




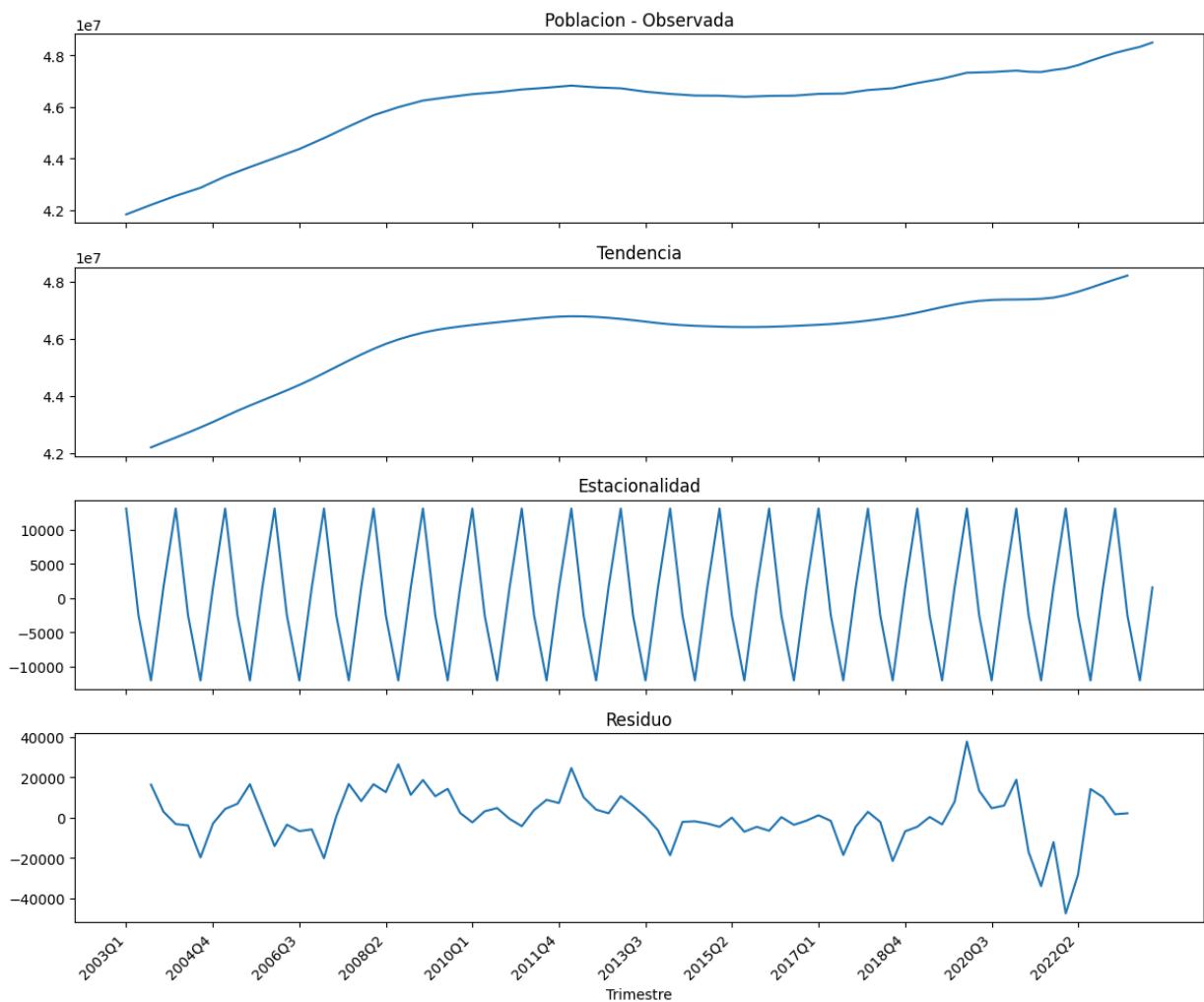
Pib_Trimestral_Ajustado - Observada

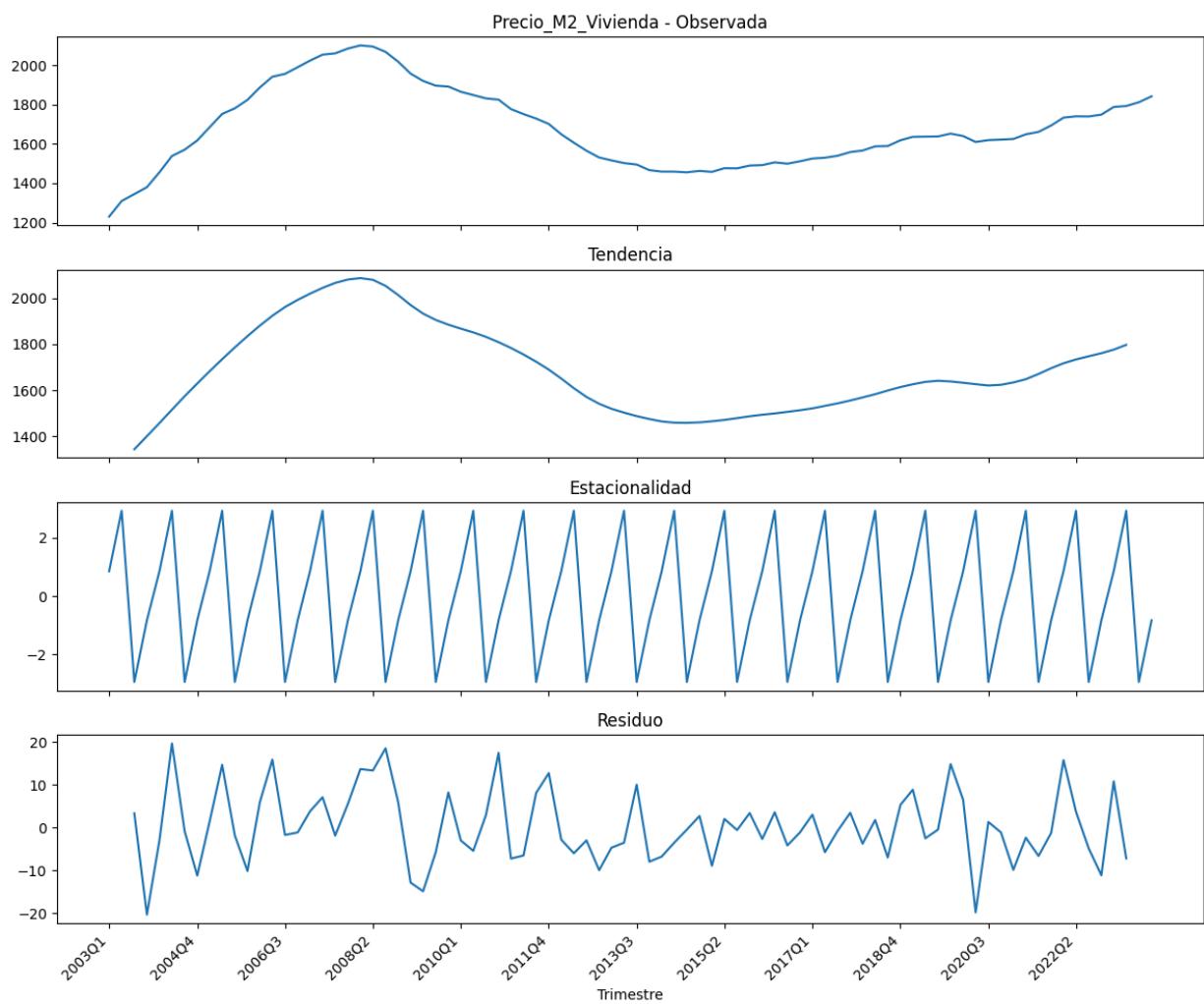


--- Descomposición de la serie temporal: Pib_Trimestral_No_Ajustado ---

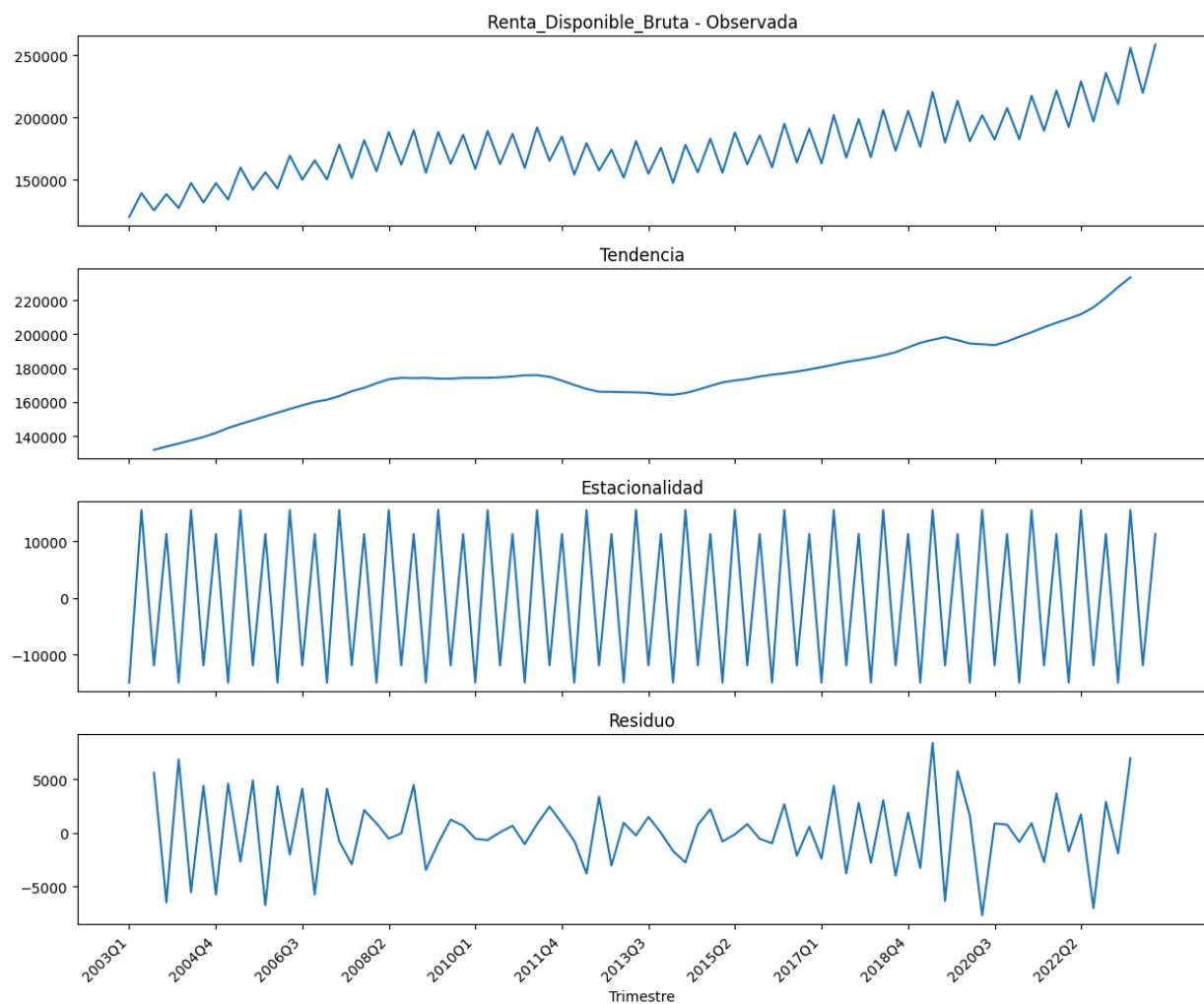


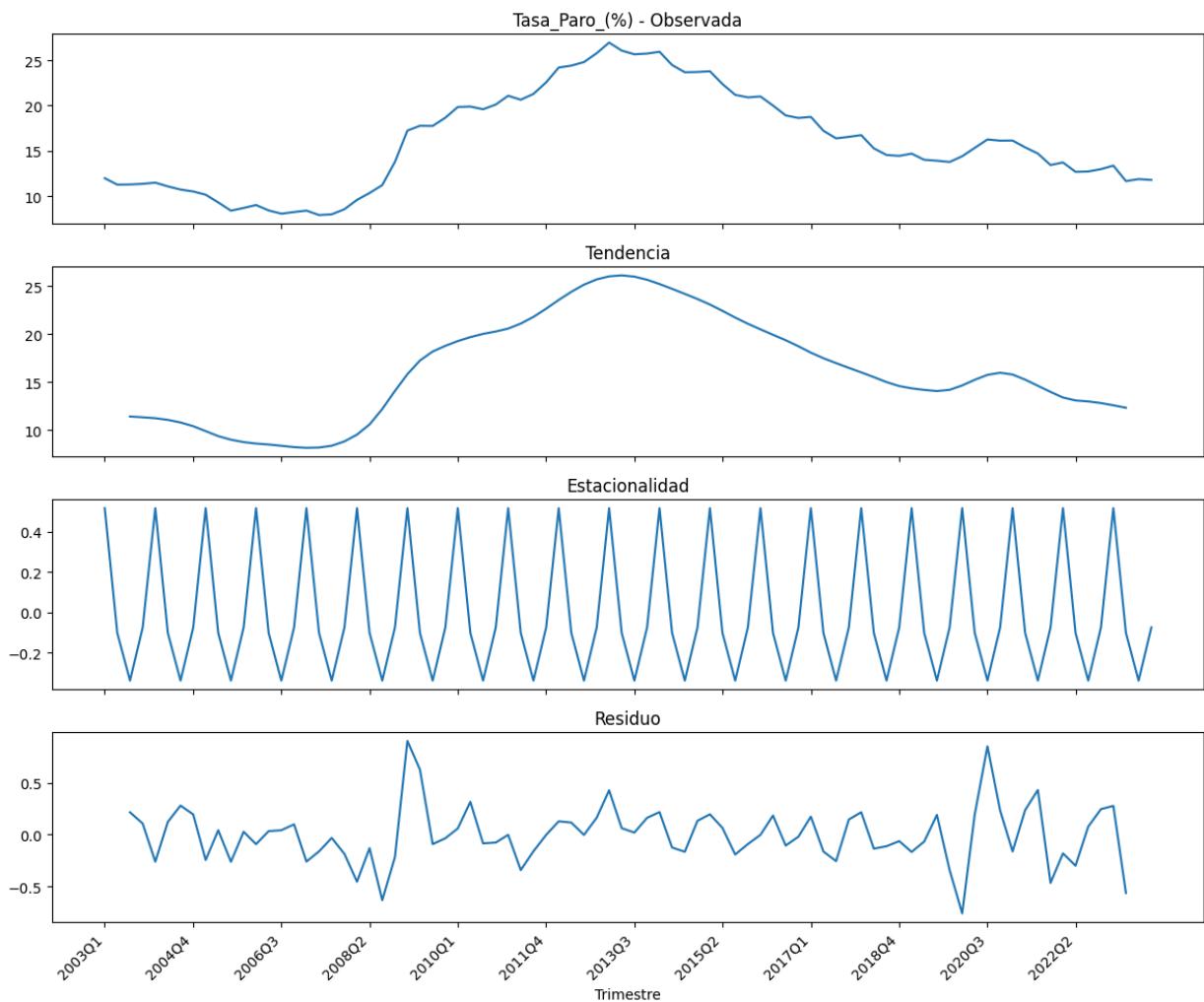
--- Descomposición de la serie temporal: Poblacion ---

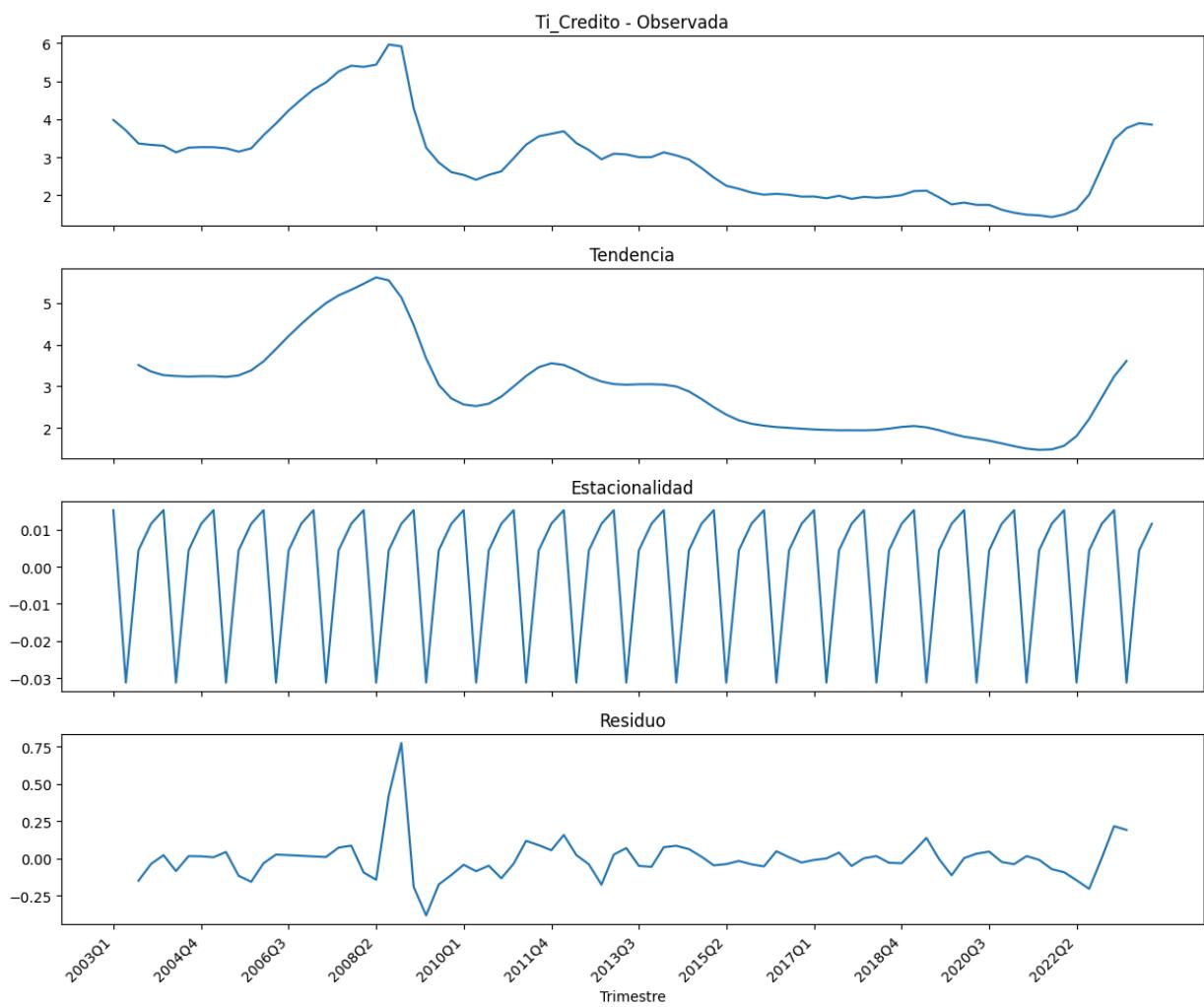




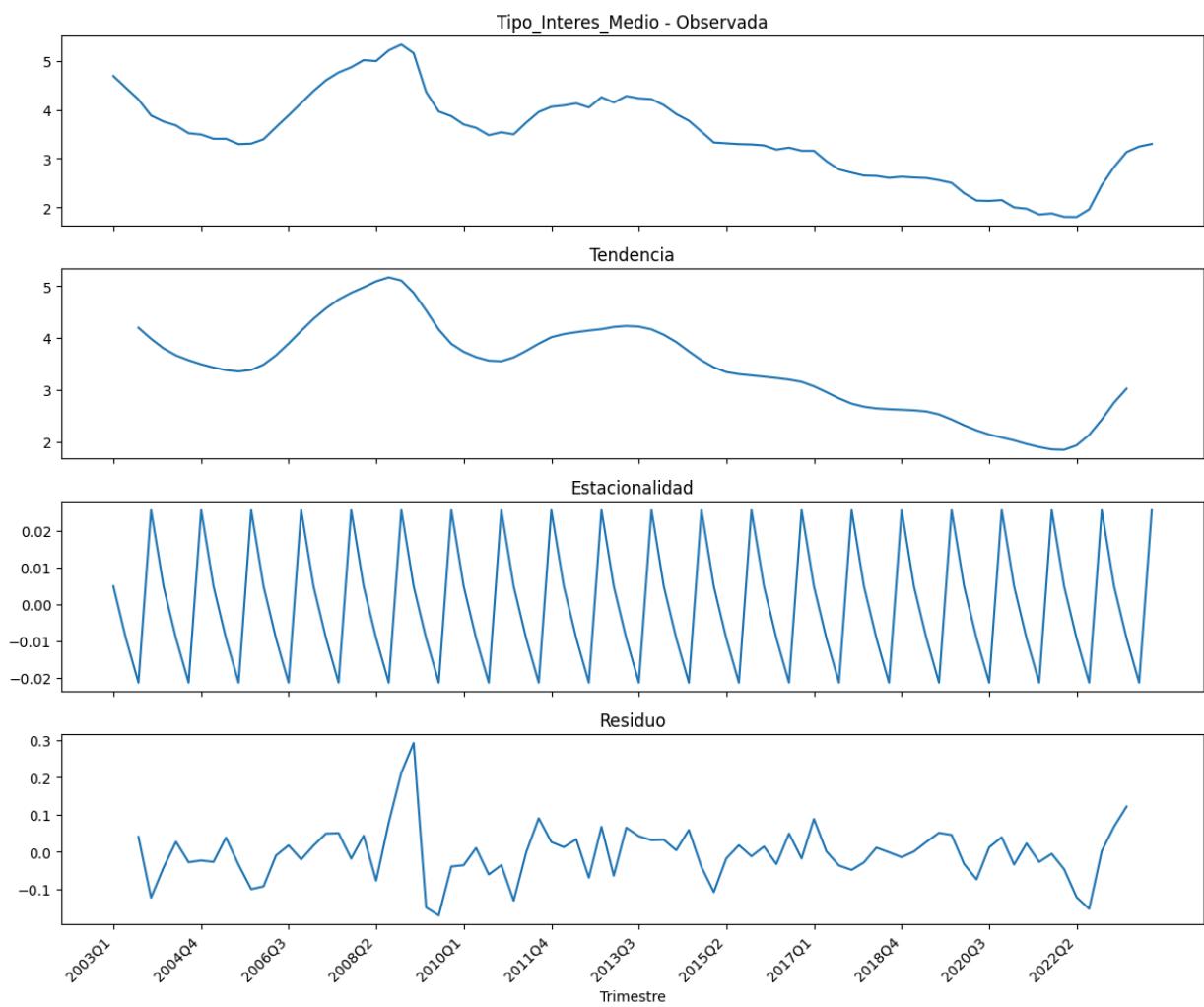
--- Descomposición de la serie temporal: Renta_Disponible_Bruta ---



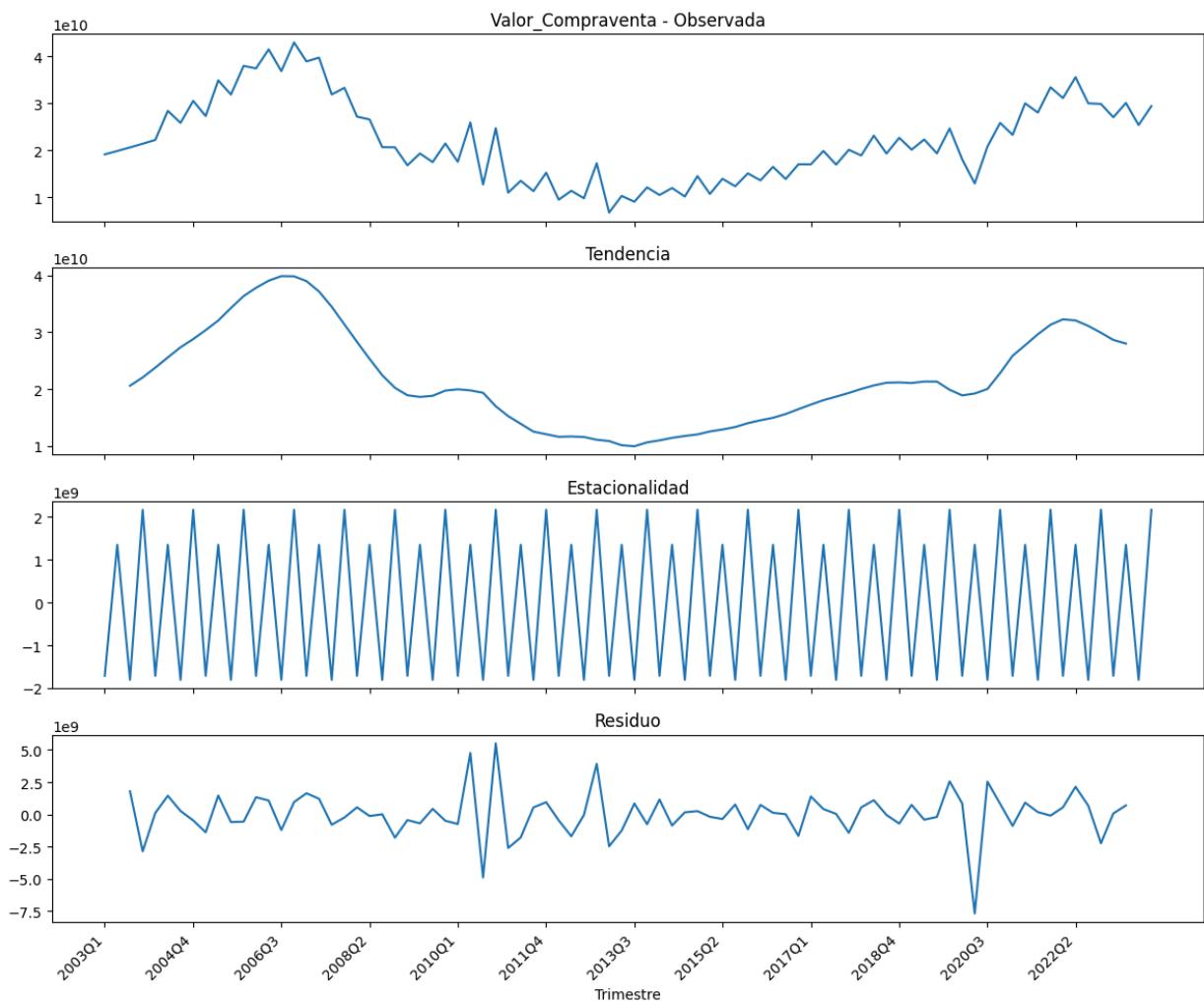


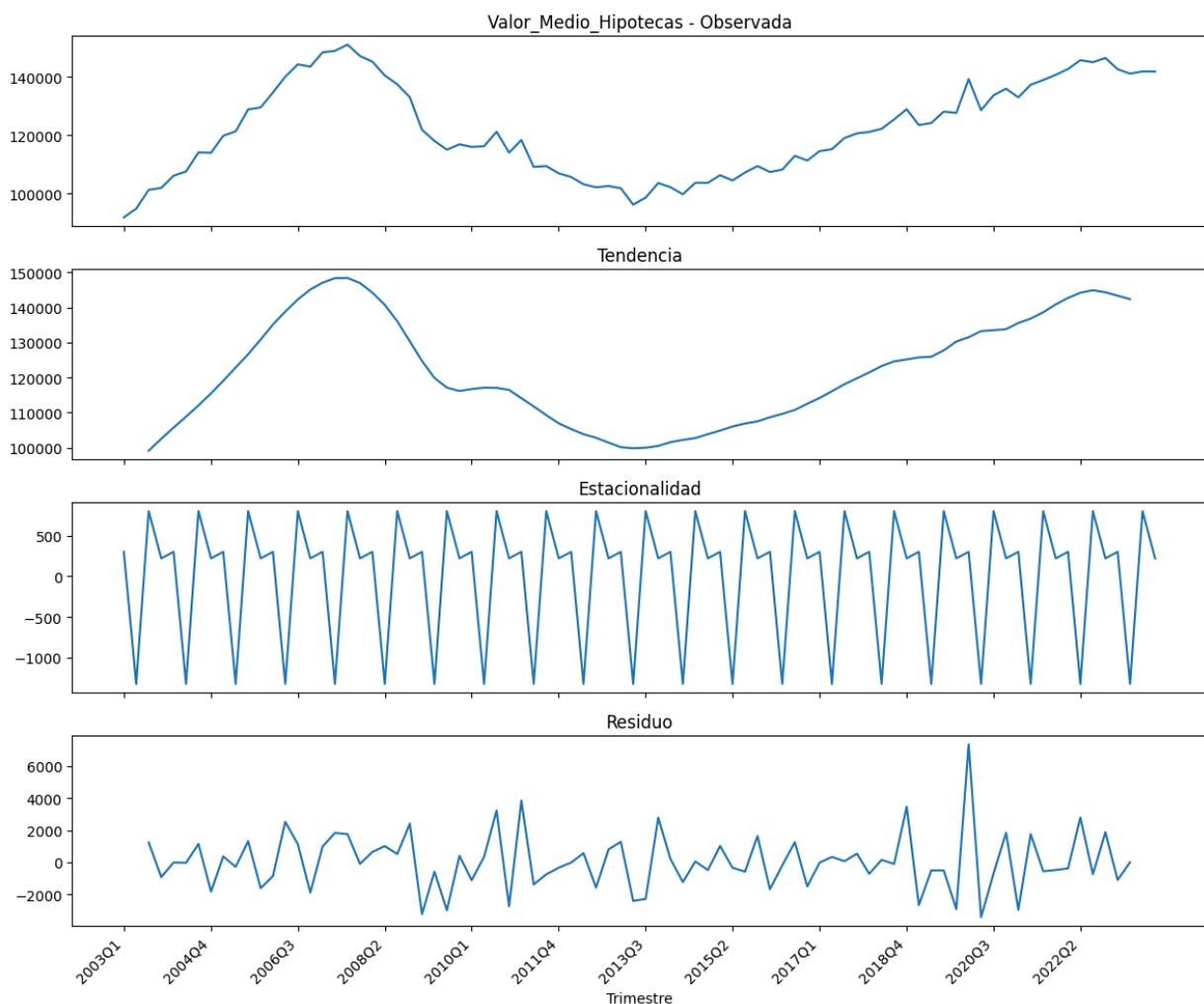


--- Descomposición de la serie temporal: Tipo_Interes_Medio ---



--- Descomposición de la serie temporal: Valor_Compraventa ---





Análisis de estacionalidad y método ANOVA

En este caso, vamos a agrupar las variables por trimestres para realizar un análisis estacional. Lo que vamos a ver en cada gráfico es:

- **Mediana (línea central de la caja):** es el valor típico de la variable para ese trimestre
- **Caja (desde el primer al tercer cuartil):** es el rango donde está el 50% de los datos centrales
- **Bigotes y puntos fuera (outliers):** variabilidad y valores extremos

Lo que nos va a permitir es una comparación entre trimestres. Es decir, si la variable tiene valores más altos, más dispersos o más atípicos en ciertos trimestres

- **Si las cajas son similares entre trimestres,** podemos decir que la variable no tiene mucha estacionalidad
- **Si las medianas cambian notablemente entre trimestres,** puede ser un indicio de una estacionalidad clara
- **Si la dispersión o presencia de outliers cambia mucho,** puede indicar que en ciertos trimestres hay más volatilidad o eventos especiales

```
In [14]: df_clean['Trimestre_simple'] = df_clean['Fecha'].dt.quarter.astype(str).replace({'1': 'Q1', '2': 'Q2', '3': 'Q3'})

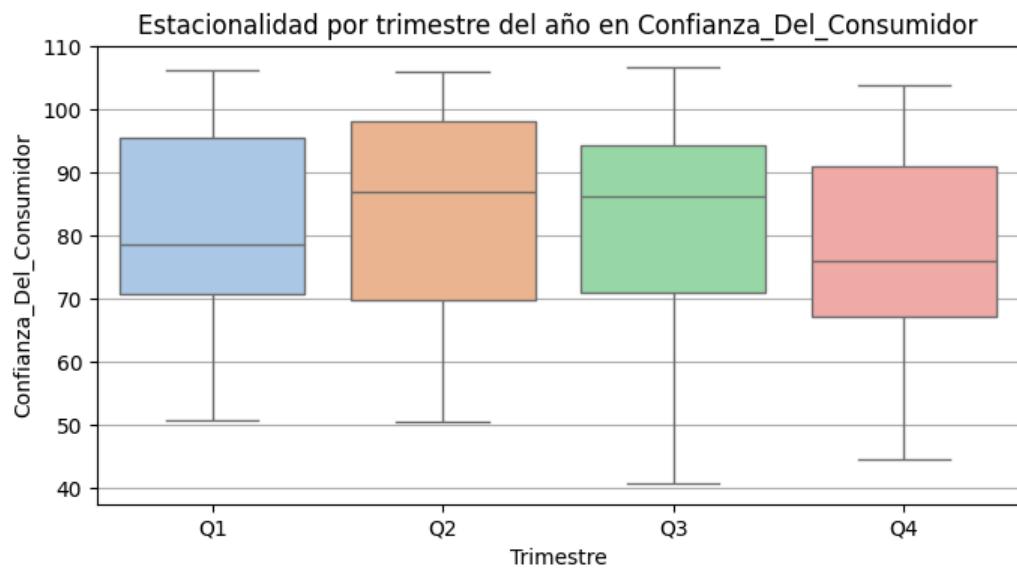
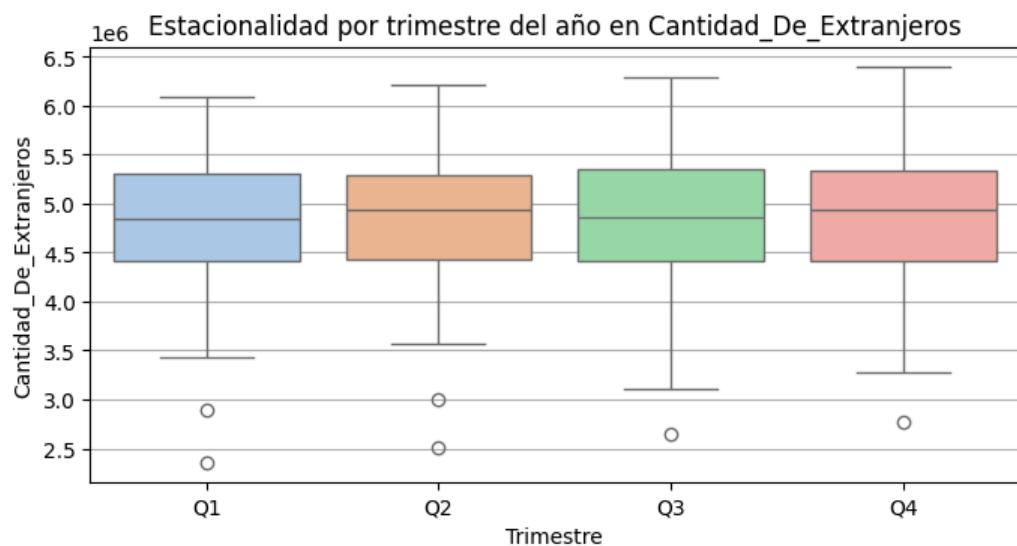
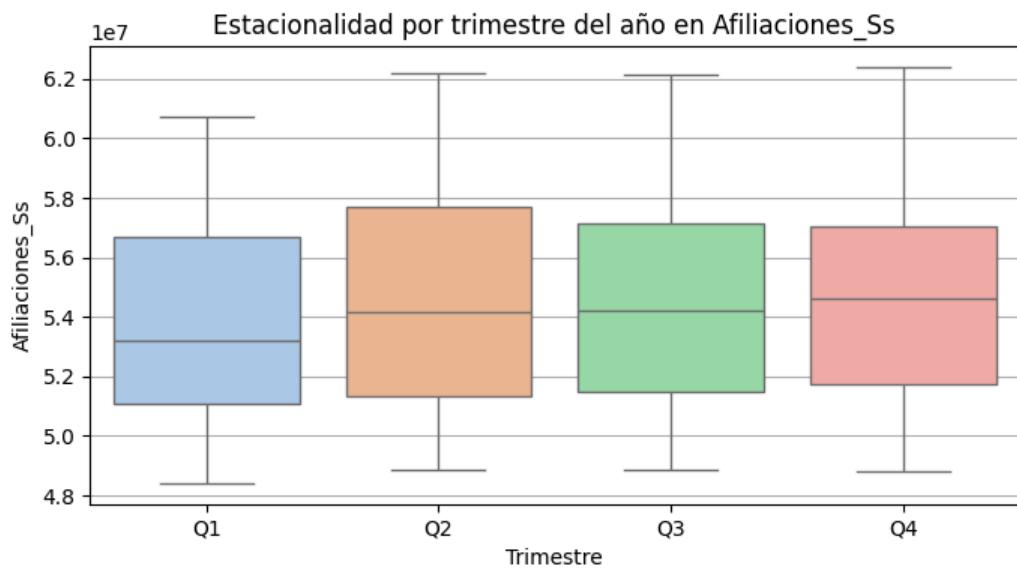
cols = df_clean.select_dtypes(include=['float64', 'int64']).columns.difference(['Fecha_ordinal'])

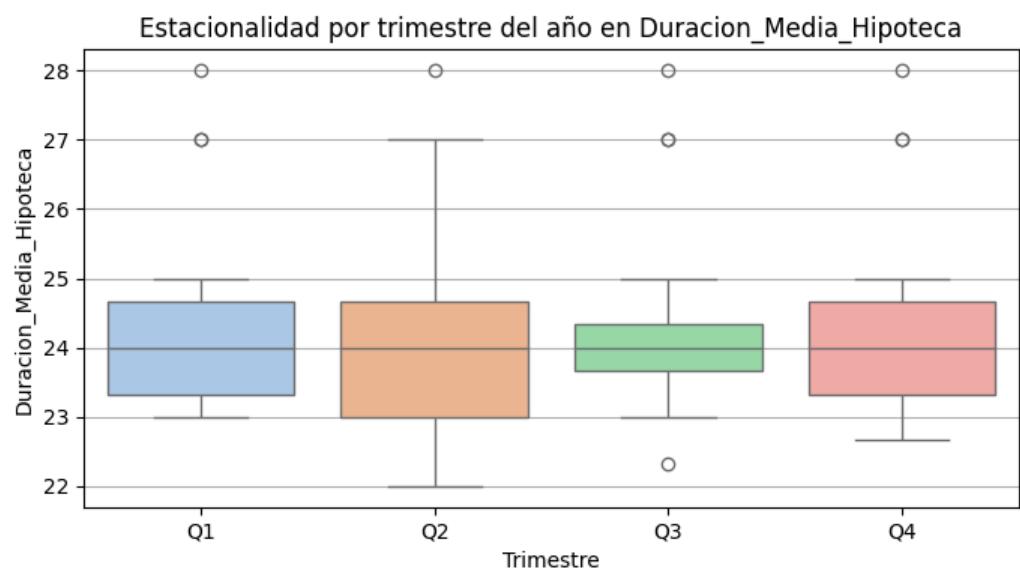
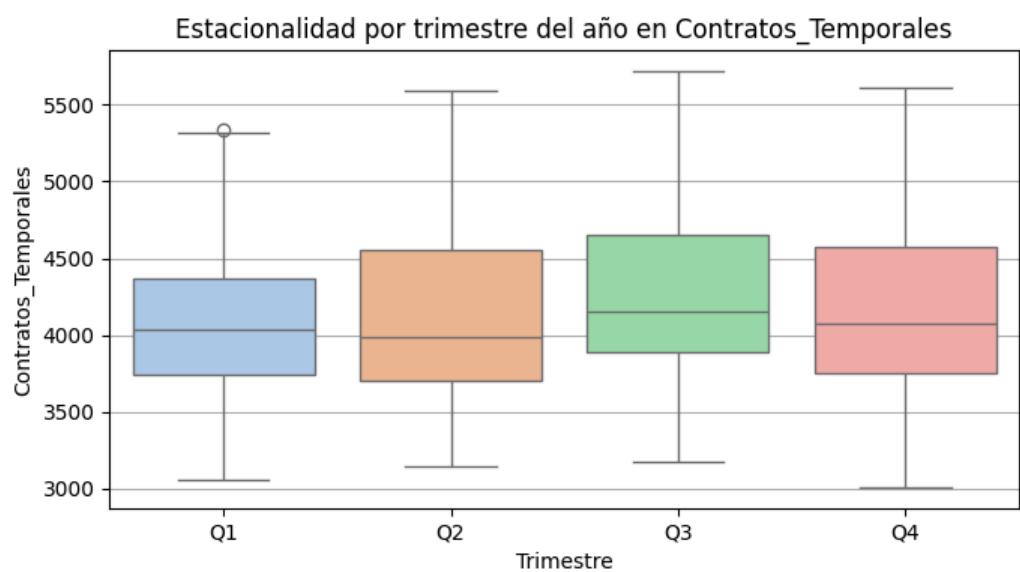
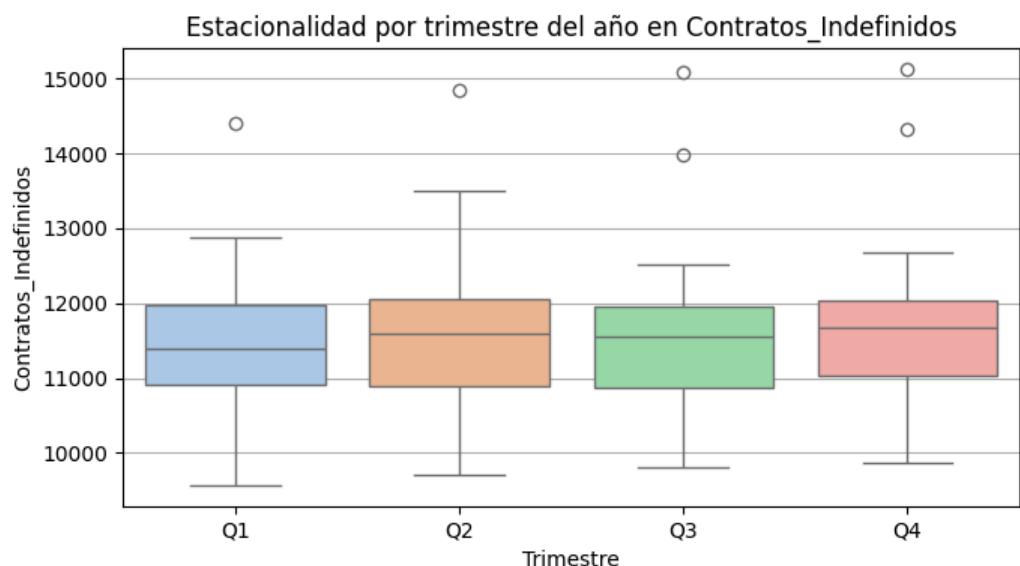
for col in cols:
    plt.figure(figsize=(7, 4))
    sns.boxplot(
        x='Trimestre_simple',
        y=col,
        hue='Trimestre_simple',
        data=df_clean,
        palette='pastel',
        legend=False
    )
```

```

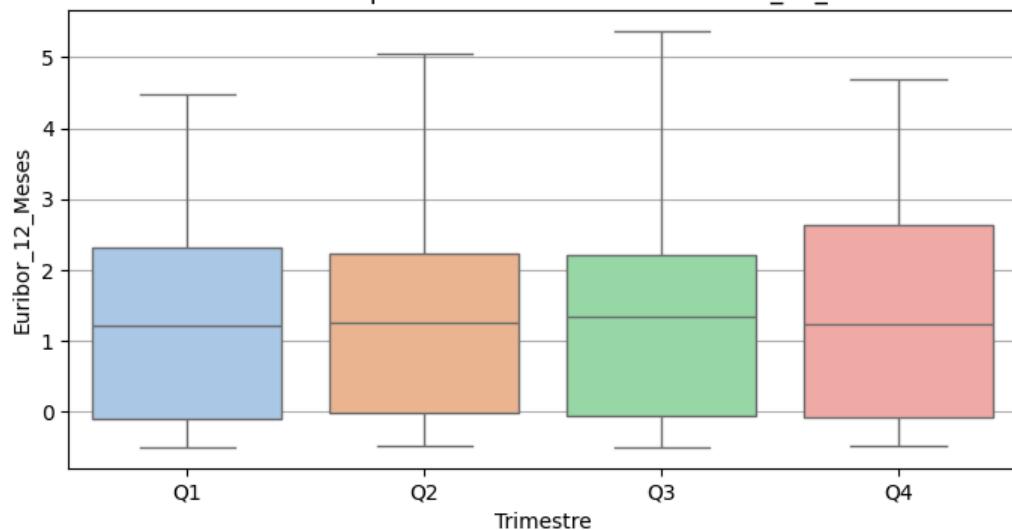
plt.title(f'Estacionalidad por trimestre del año en {col}')
plt.xlabel('Trimestre')
plt.ylabel(col)
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```

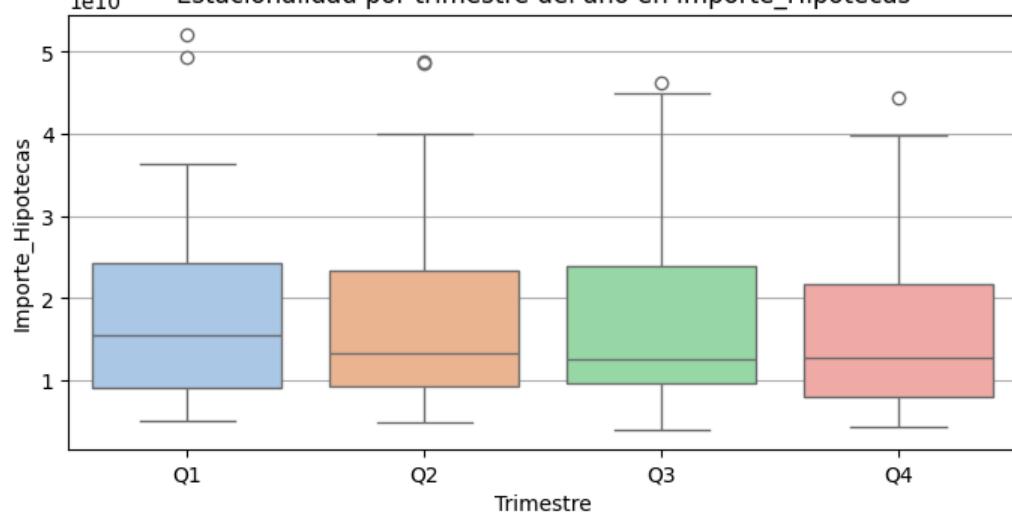




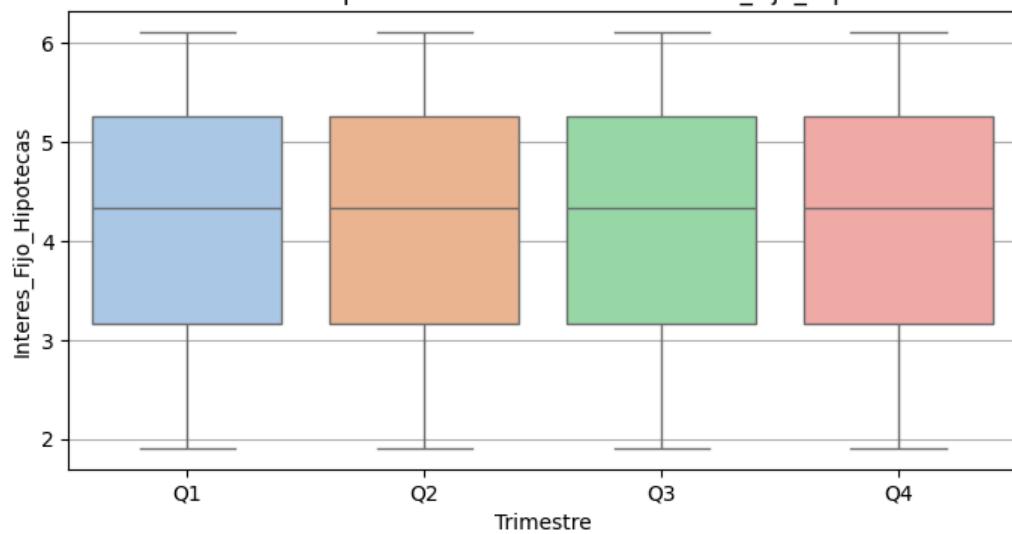
Estacionalidad por trimestre del año en Euribor_12_Meses



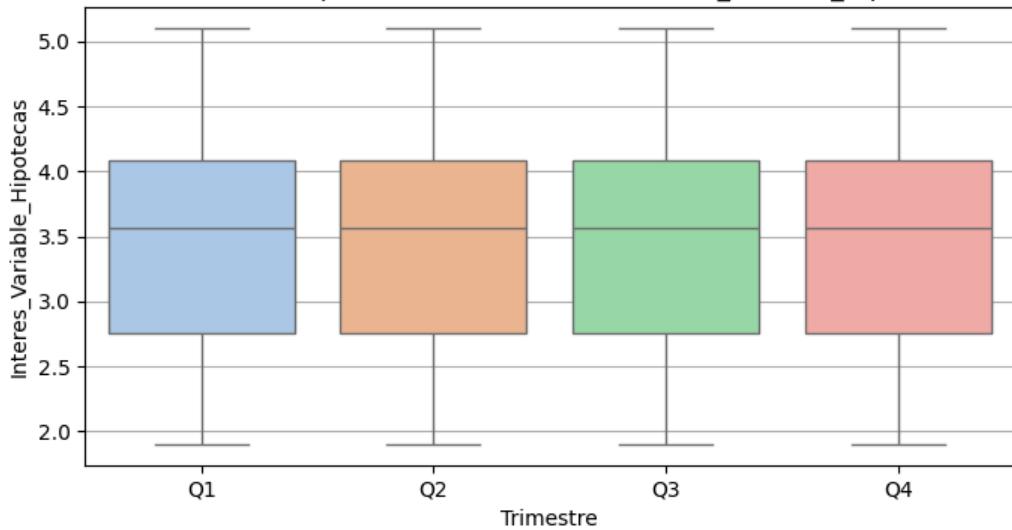
Estacionalidad por trimestre del año en Importe_Hipotecas



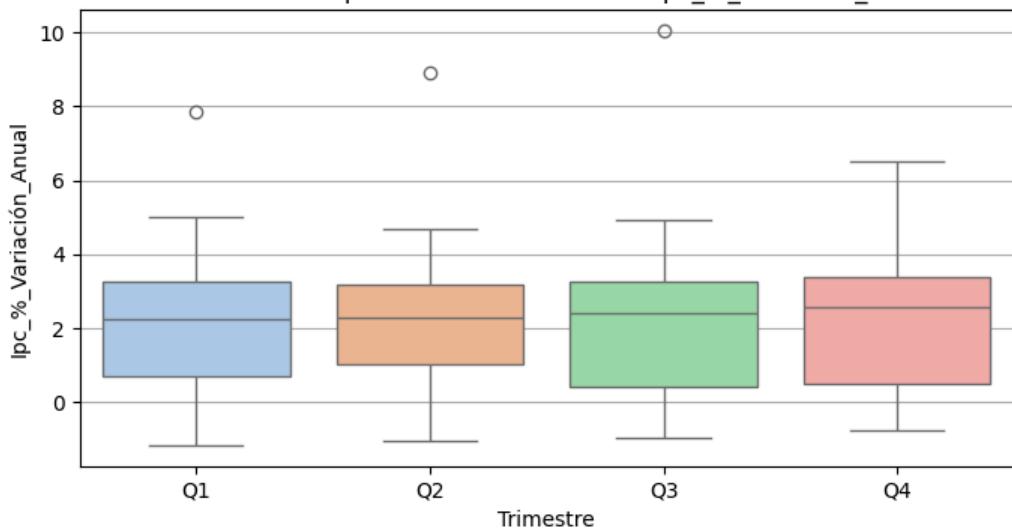
Estacionalidad por trimestre del año en Interes_Fijo_Hipotecas



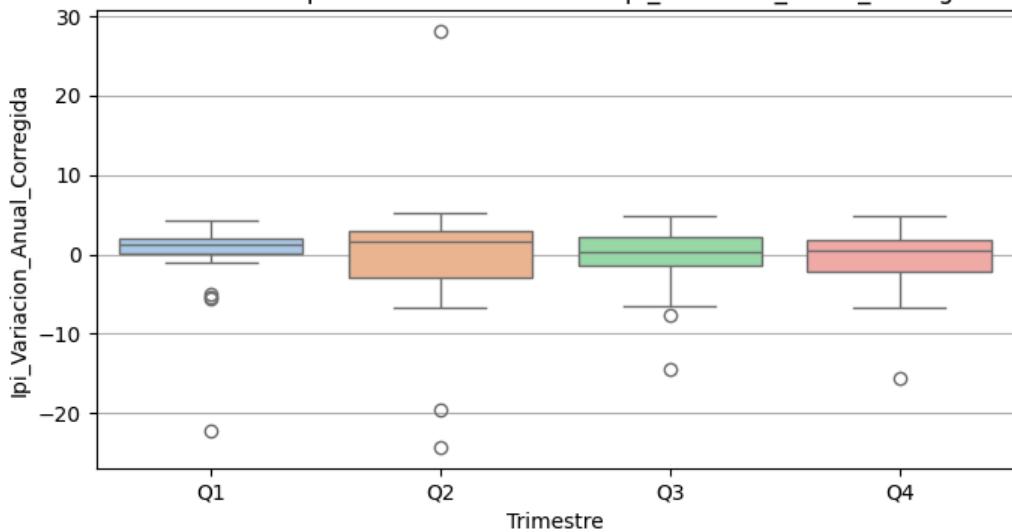
Estacionalidad por trimestre del año en Interes_Variable_Hipotecas



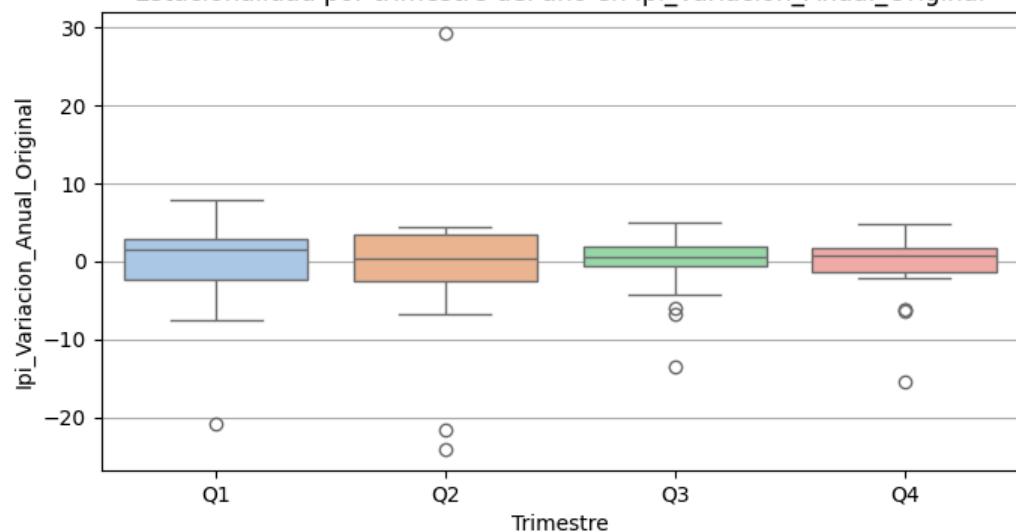
Estacionalidad por trimestre del año en Ipc_%_Variación_Anual



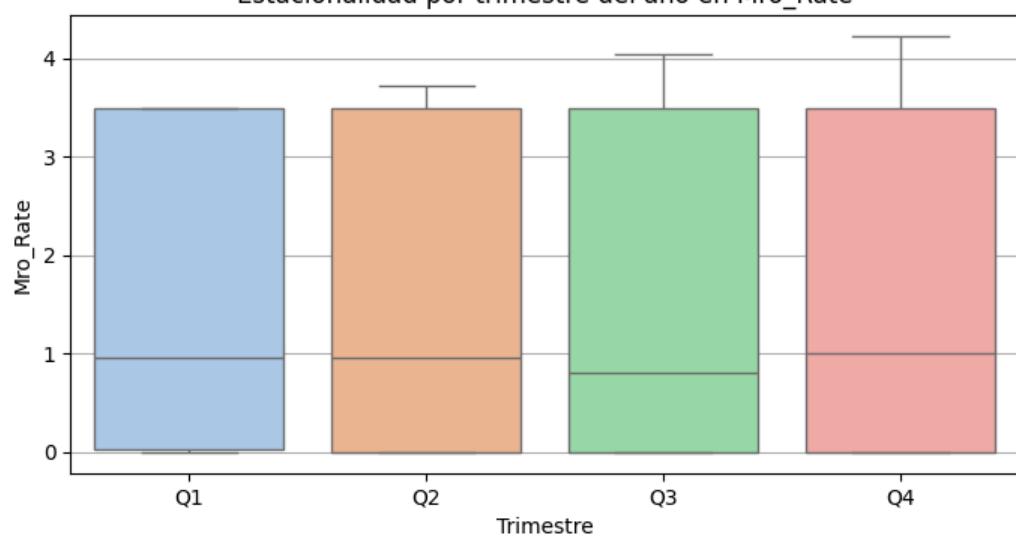
Estacionalidad por trimestre del año en Ipi_Variacion_Anual_Corregida



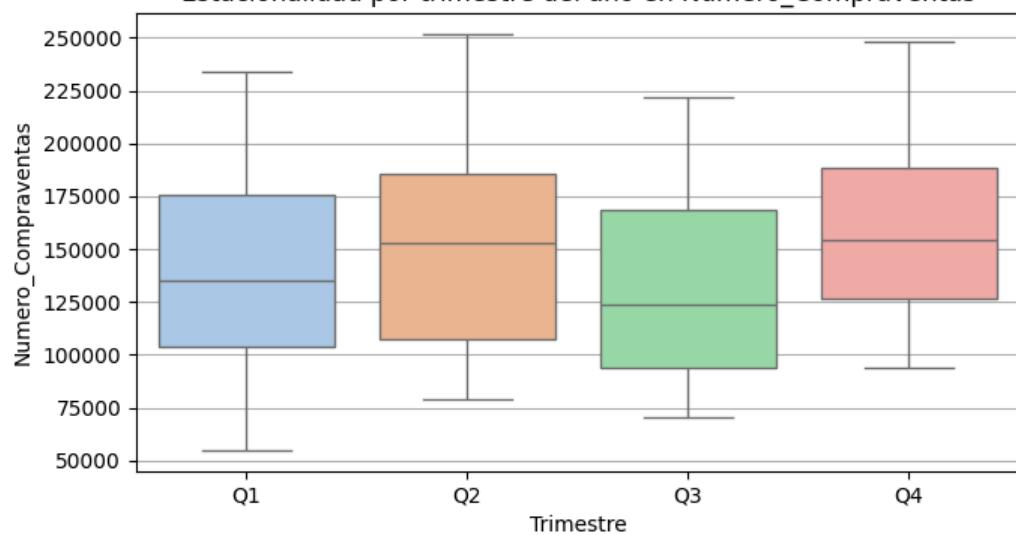
Estacionalidad por trimestre del año en Ipi_Variacion_Anual_Original



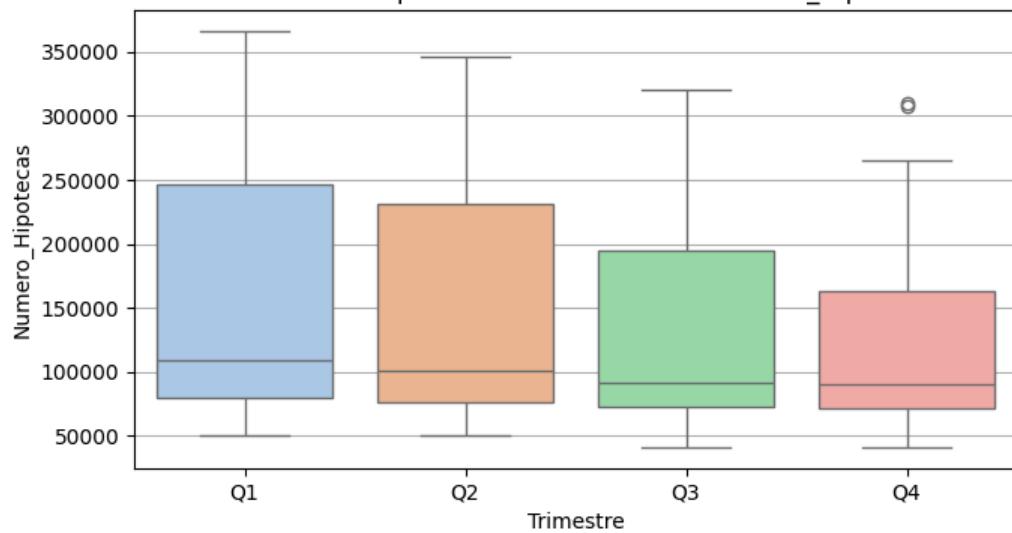
Estacionalidad por trimestre del año en Mro_Rate



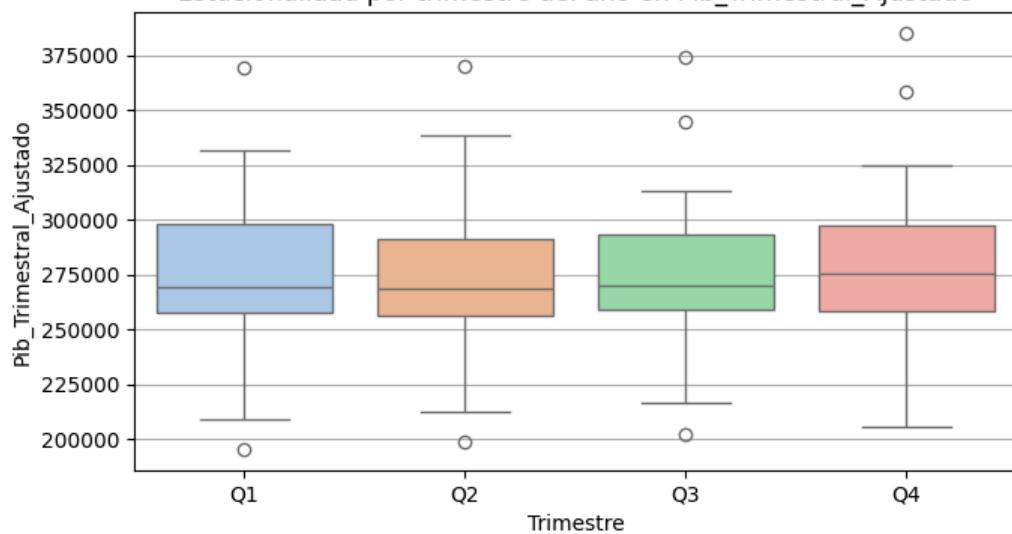
Estacionalidad por trimestre del año en Numero_Compraventas



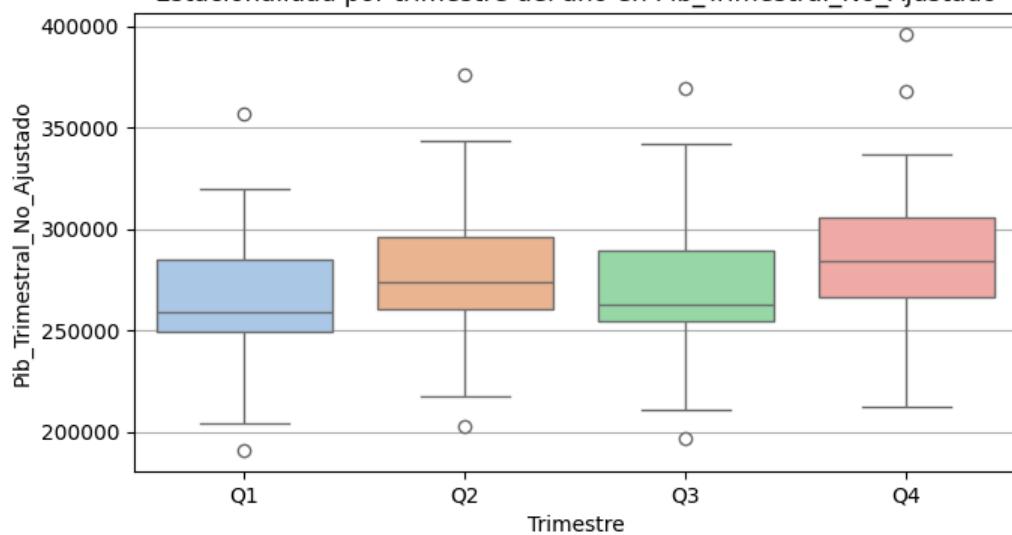
Estacionalidad por trimestre del año en Numero_Hipotecas

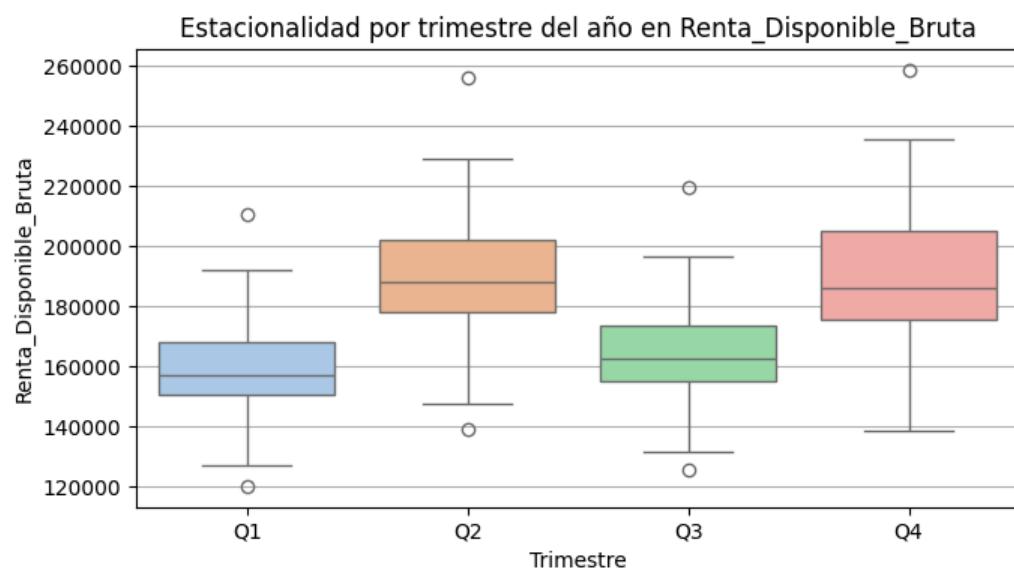
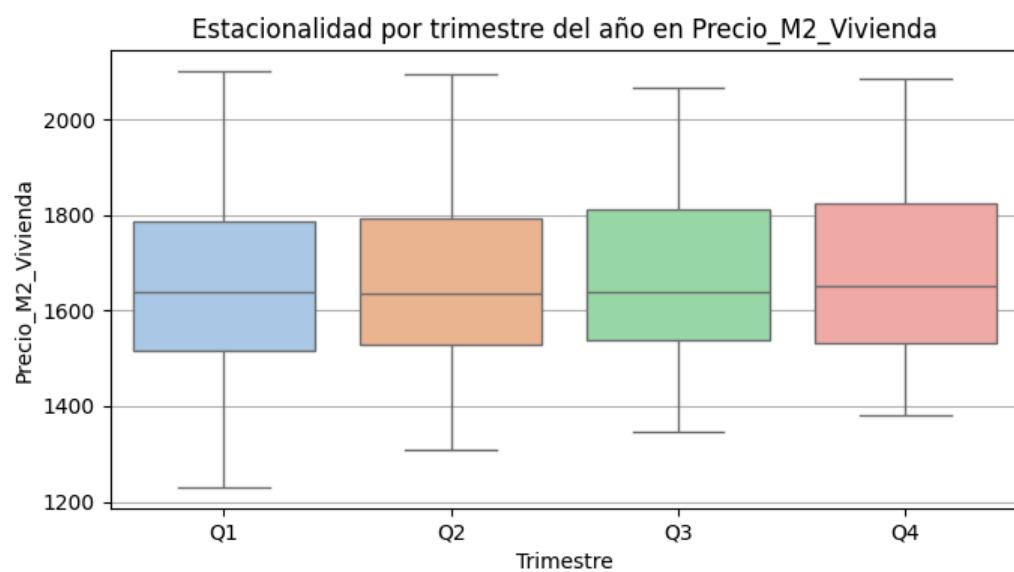
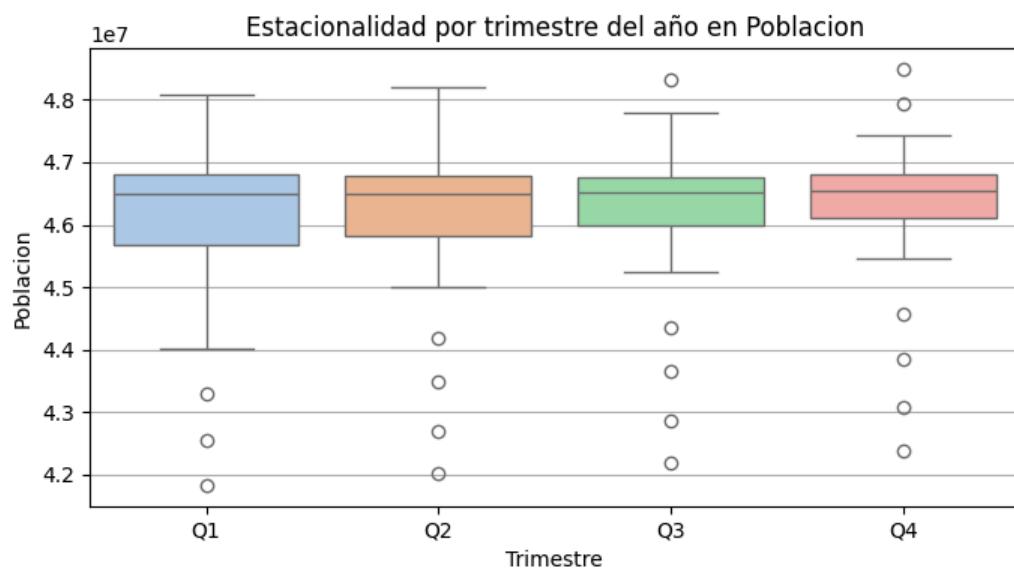


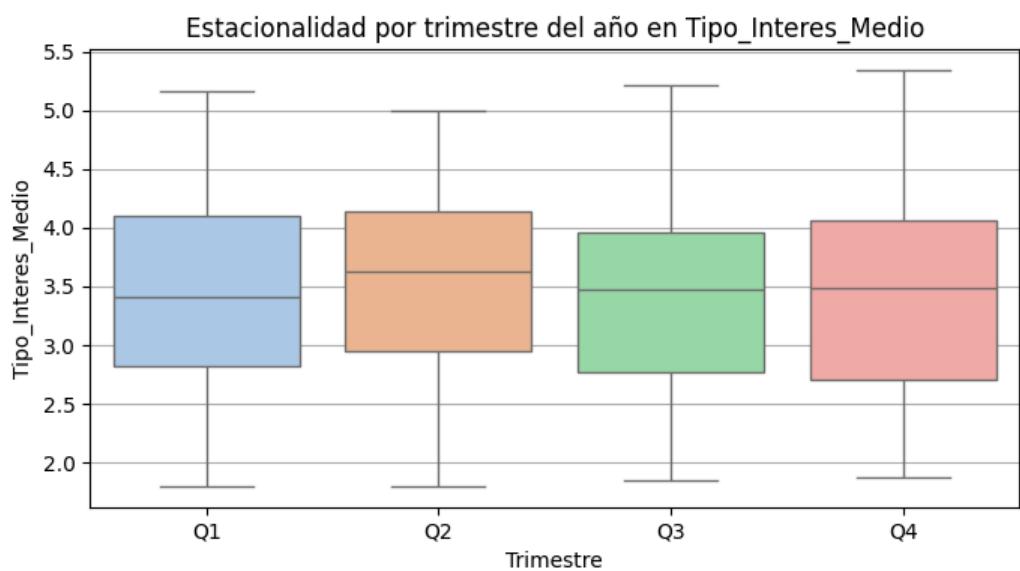
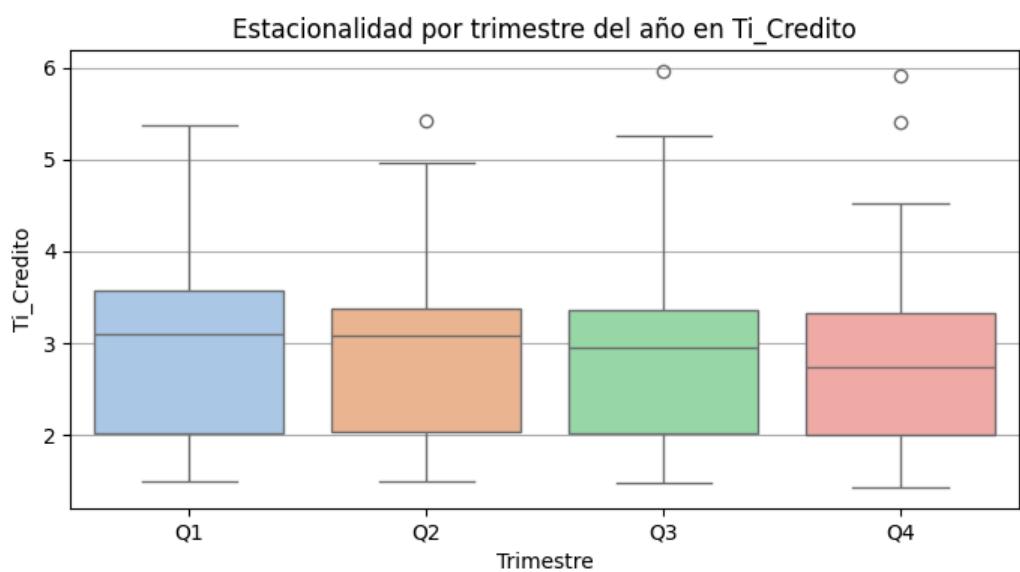
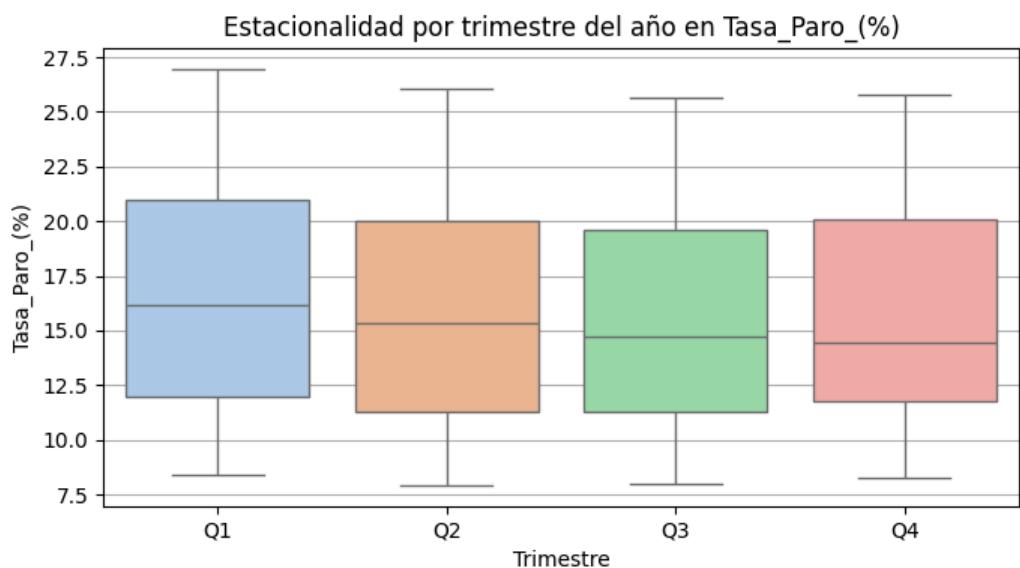
Estacionalidad por trimestre del año en Pib_Trimestral_Ajustado

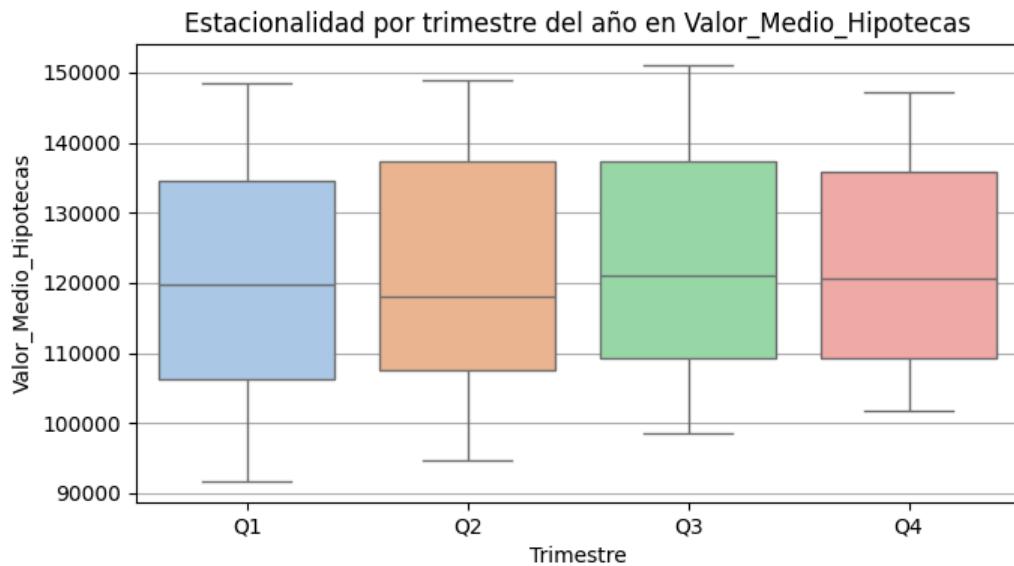
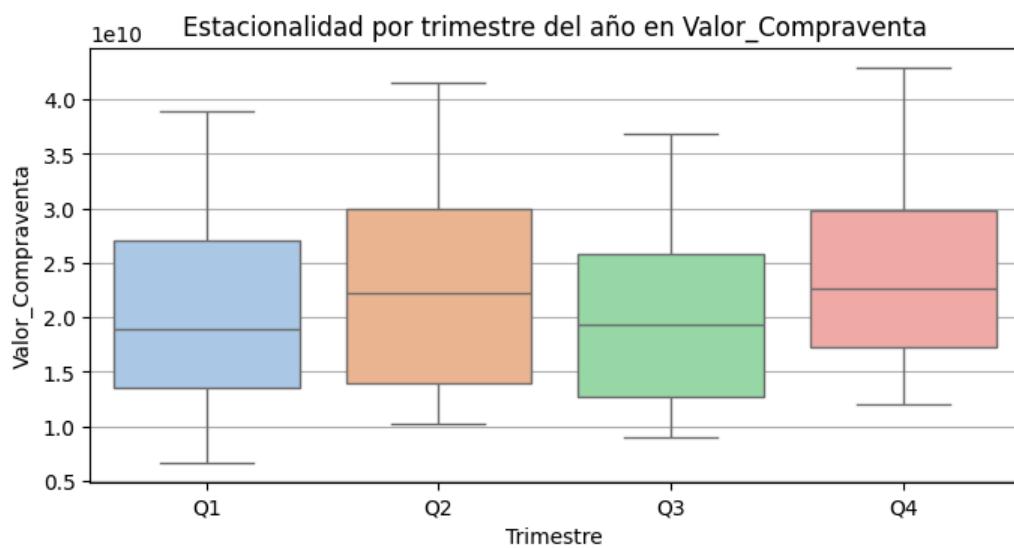


Estacionalidad por trimestre del año en Pib_Trimestral_No_Ajustado









A continuación, vamos a estudiar también si hay diferencias significativas entre trimestres por cada variable con el **método ANOVA (Analysis of Variance)**, que es una técnica estadística que nos ayuda a determinar si al menos alguno de los grupos/trimestres tiene una media significativamente distinta a las demás. En este sentido, si detectamos diferencias significativas en las medias de una variable según el trimestre, es una señal clara de que la variable presenta estacionalidad

Para ello, con el objetivo de hacerlo con el mayor rigor estadístico posible, dividimos las variables en tres grupos:

- **1. Variables totales acumuladas o sumatorios absolutos:** tenemos que trabajar con tasas de crecimiento interanuales o trimestrales

Afiliaciones_Ss (total personas afiliadas)

Cantidad_De_Extranjeros (total población extranjera)

Contratos_Indefinidos (total contratos indefinidos)

Contratos_Temporales (total contratos temporales)

Importe_Hipotecas (€ total importe hipotecas)

Numero_Compraventas (total compraventas vivienda)

Numero_Hipotecas (total hipotecas concedidas)

Poblacion (total población trimestral, acumulativa)

Renta_Disponible_Bruta (€ total renta disponible bruta)

Valor_Compraventa (€ total valor compraventas)

Valor_Medio_Hipotecas (€ valor medio hipotecas)

- **2. Variables índices, tasas o porcentajes:** podemos trabajar directamente con estas variables sin ninguna transformación

Confianza_Del_Consumidor (índice)
 Euribor_12_Meses (%)
 Interes_Fijo_Hipotecas (%)
 Interes_Variable_Hipotecas (%)
 Ipc_%_Variación_Anual (%)
 Ipi_Variacion_Anual_Corregida (%)
 Ipi_Variacion_Anual_Original (%)
 Mro_Rate (%)
 Pib_Trimestral_Ajustado (%)
 Pib_Trimestral_No_Ajustado (%)
 Tasa_Paro_(%) (%)
 Ti_Credito (%)
 Tipo_Interes_Medio (%)

- **3. Variables de duración o medias:** aplicamos el test ANOVA directamente sin transformación

Duracion_Media_Hipoteca (años) Precio_M2_Vivienda (€ precio medio vivienda por m2)

```
In [15]: import pandas as pd
from scipy.stats import f_oneway

# 1. Añadir columnas Año y Trimestre_simple si no están
df_clean['Año'] = df_clean['Fecha'].dt.year
df_clean['Trimestre_simple'] = df_clean['Fecha'].dt.quarter.map({1:'Q1', 2:'Q2', 3:'Q3', 4:'Q4'})

# 2. Variables según tipo
variables_totales = [
    'Afiliaciones_Ss',
    'Cantidad_De_Extranjeros',
    'Contratos_Indefinidos',
    'Contratos_Temporales',
    'Importe_Hipotecas',
    'Numero_Compraventas',
    'Numero_Hipotecas',
    'Poblacion',
    'Renta_Disponible_Bruta',
    'Valor_Compraventa',
    'Valor_Medio_Hipotecas'
]

variables_medias = [
    'Duracion_Media_Hipoteca',
    'Precio_M2_Vivienda'
]

variables_indices = [
    'Confianza_Del_Consumidor',
    'Euribor_12_Meses',
    'Interes_Fijo_Hipotecas',
    'Interes_Variable_Hipotecas',
    'Ipc_%_Variación_Anual',
    'Ipi_Variacion_Anual_Corregida',
    'Ipi_Variacion_Anual_Original',
    'Mro_Rate',
    'Pib_Trimestral_Ajustado',
    'Pib_Trimestral_No_Ajustado',
    'Tasa_Paro_(%)',
    'Ti_Credito',
    'Tipo_Interes_Medio'
]

# 3. Calcular variación interanual % para variables totales (por trimestre)
df_variacion = df_clean.copy()
```

```

for var in variables_totales:
    df_variacion[var + '_var_interanual'] = df_variacion.groupby('Trimestre_simple')[var].pct_change() * 100

# 4. ANOVA variables indices y medias (valores directos)
print("ANOVA variables índices y medias (valores directos):\n")

for var in variables_indices + variables_medias:
    grupos = [df_clean[df_clean['Trimestre_simple'] == t][var].dropna() for t in ['Q1', 'Q2', 'Q3', 'Q4']]
    grupos = [g for g in grupos if len(g) > 0]

    if len(grupos) > 1:
        stat, p = f_oneway(*grupos)
        print(f'{var}: F={stat:.3f}, p={p:.4f}')
        if p < 0.05:
            print(" -> Diferencias significativas entre trimestres\n")
        else:
            print(" -> No hay diferencias significativas entre trimestres\n")
    else:
        print(f'{var}: No hay suficientes datos para ANOVA\n')

# 5. ANOVA variables totales (con variaciones interanuales)
print("ANOVA variables totales (variación interanual %):\n")

for var in variables_totales:
    var_var = var + '_var_interanual'
    grupos = [df_variacion[df_variacion['Trimestre_simple'] == t][var_var].dropna() for t in ['Q1', 'Q2', 'Q3', 'Q4']]
    grupos = [g for g in grupos if len(g) > 0]

    if len(grupos) > 1:
        stat, p = f_oneway(*grupos)
        print(f'{var} (var interanual): F={stat:.3f}, p={p:.4f}')
        if p < 0.05:
            print(" -> Diferencias significativas entre trimestres\n")
        else:
            print(" -> No hay diferencias significativas entre trimestres\n")
    else:
        print(f'{var} (var interanual): No hay suficientes datos para ANOVA\n')

```

ANOVA variables indices y medias (valores directos):

Confianza_Del_Consumidor: F=0.215, p=0.8858
-> No hay diferencias significativas entre trimestres

Euribor_12_Meses: F=0.032, p=0.9923
-> No hay diferencias significativas entre trimestres

Interes_Fijo_Hipotecas: F=-0.000, p=nan
-> No hay diferencias significativas entre trimestres

Interes_Variable_Hipotecas: F=0.000, p=1.0000
-> No hay diferencias significativas entre trimestres

Ipc_%_Variación_Anual: F=0.003, p=0.9997
-> No hay diferencias significativas entre trimestres

Ipi_Variacion_Anual_Corregida: F=0.030, p=0.9929
-> No hay diferencias significativas entre trimestres

Ipi_Variacion_Anual_Original: F=0.036, p=0.9906
-> No hay diferencias significativas entre trimestres

Mro_Rate: F=0.003, p=0.9998
-> No hay diferencias significativas entre trimestres

Pib_Trimestral_Ajustado: F=0.170, p=0.9165
-> No hay diferencias significativas entre trimestres

Pib_Trimestral_No_Ajustado: F=1.557, p=0.2062
-> No hay diferencias significativas entre trimestres

Tasa_Paro_(%): F=0.086, p=0.9675
-> No hay diferencias significativas entre trimestres

Ti_Credito: F=0.008, p=0.9990
-> No hay diferencias significativas entre trimestres

Tipo_Interes_Medio: F=0.016, p=0.9971
-> No hay diferencias significativas entre trimestres

Duracion_Media_Hipoteca: F=0.012, p=0.9982
-> No hay diferencias significativas entre trimestres

Precio_M2_Vivienda: F=0.040, p=0.9893
-> No hay diferencias significativas entre trimestres

ANOVA variables totales (variación interanual %):

Afiliaciones_Ss (var interanual): F=0.000, p=1.0000
-> No hay diferencias significativas entre trimestres

Cantidad_De_Extranjeros (var interanual): F=0.027, p=0.9940
-> No hay diferencias significativas entre trimestres

Contratos_Indefinidos (var interanual): F=0.003, p=0.9997
-> No hay diferencias significativas entre trimestres

Contratos_Temporales (var interanual): F=0.009, p=0.9987
-> No hay diferencias significativas entre trimestres

Importe_Hipotecas (var interanual): F=0.013, p=0.9979
-> No hay diferencias significativas entre trimestres

Numero_Compraventas (var interanual): F=0.132, p=0.9406
-> No hay diferencias significativas entre trimestres

Numero_Hipotecas (var interanual): F=0.007, p=0.9992
-> No hay diferencias significativas entre trimestres

Poblacion (var interanual): F=0.004, p=0.9996
-> No hay diferencias significativas entre trimestres

Renta_Disponible_Bruta (var interanual): F=0.042, p=0.9883
-> No hay diferencias significativas entre trimestres

Valor_Compraventa (var interanual): F=0.122, p=0.9467
-> No hay diferencias significativas entre trimestres

Valor_Medio_Hipotecas (var interanual): F=0.030, p=0.9928

-> No hay diferencias significativas entre trimestres

Variación anual de las variables

A continuación, vamos a analizar para cada variable cual es su **% de variación anual desde 2003 hasta 2023** y lo vamos a representar también gráficamente:

```
In [16]: df_clean['Año'] = df_clean['Fecha'].dt.year

for col in cols:
    annual_pct = df_clean.groupby('Año')[col].mean().pct_change() * 100
    annual_df = annual_pct.to_frame(name=f'Variación % {col}').round(2)

    print(f"\n📊 Porcentaje de variación anual para '{col}':")
    display(annual_df)

    plt.figure(figsize=(12, 4))
    annual_df.plot(marker='o', legend=False)
    plt.title(f'Variación anual (%) de {col}')
    plt.ylabel('% cambio')
    plt.xlabel('Año')
    plt.grid(True, axis='y', linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
```

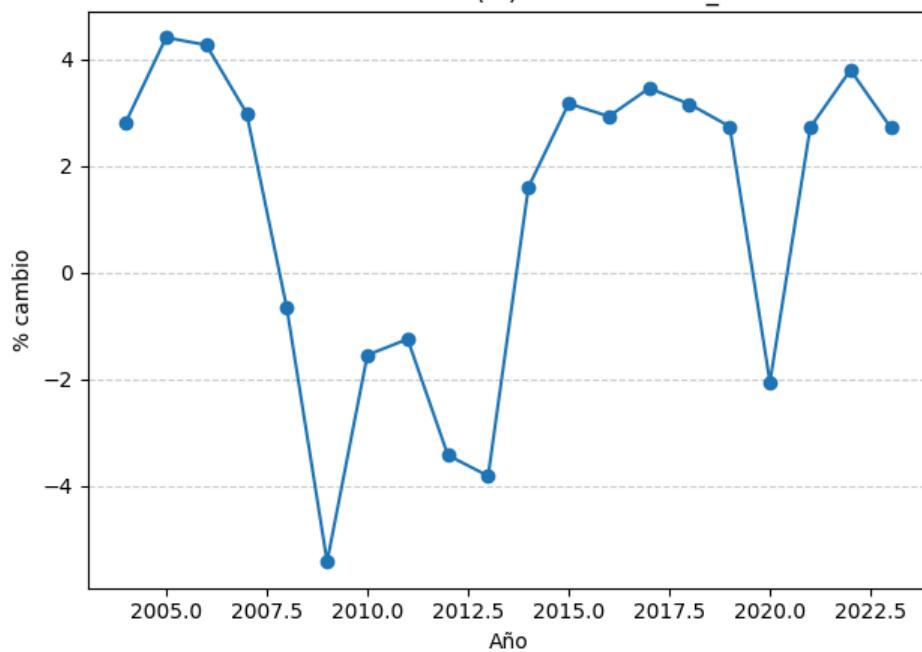
📊 Porcentaje de variación anual para 'Afiliaciones_Ss':

Variación % Afiliaciones_Ss

Año	
2003	NaN
2004	2.82
2005	4.41
2006	4.27
2007	2.99
2008	-0.67
2009	-5.43
2010	-1.55
2011	-1.25
2012	-3.43
2013	-3.82
2014	1.60
2015	3.17
2016	2.93
2017	3.46
2018	3.16
2019	2.74
2020	-2.06
2021	2.72
2022	3.80
2023	2.72

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Afiliaciones_Ss

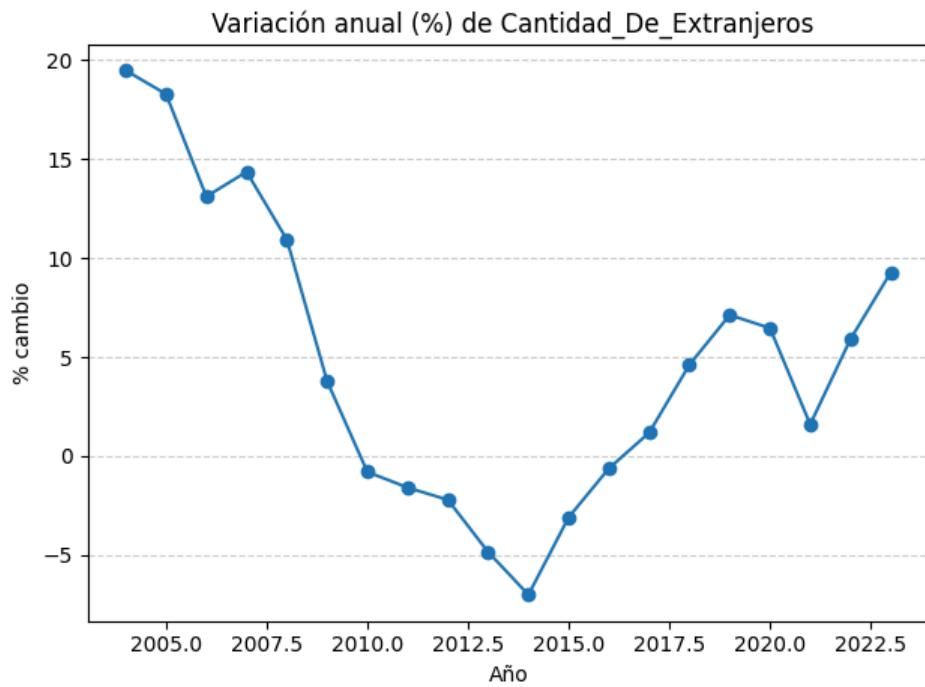


Porcentaje de variación anual para 'Cantidad_De_Extranjeros':

Variación % Cantidad_De_Extranjeros

Año	
2003	NaN
2004	19.46
2005	18.27
2006	13.10
2007	14.35
2008	10.93
2009	3.78
2010	-0.80
2011	-1.59
2012	-2.21
2013	-4.84
2014	-7.00
2015	-3.09
2016	-0.62
2017	1.18
2018	4.61
2019	7.12
2020	6.46
2021	1.60
2022	5.90
2023	9.27

<Figure size 1200x400 with 0 Axes>

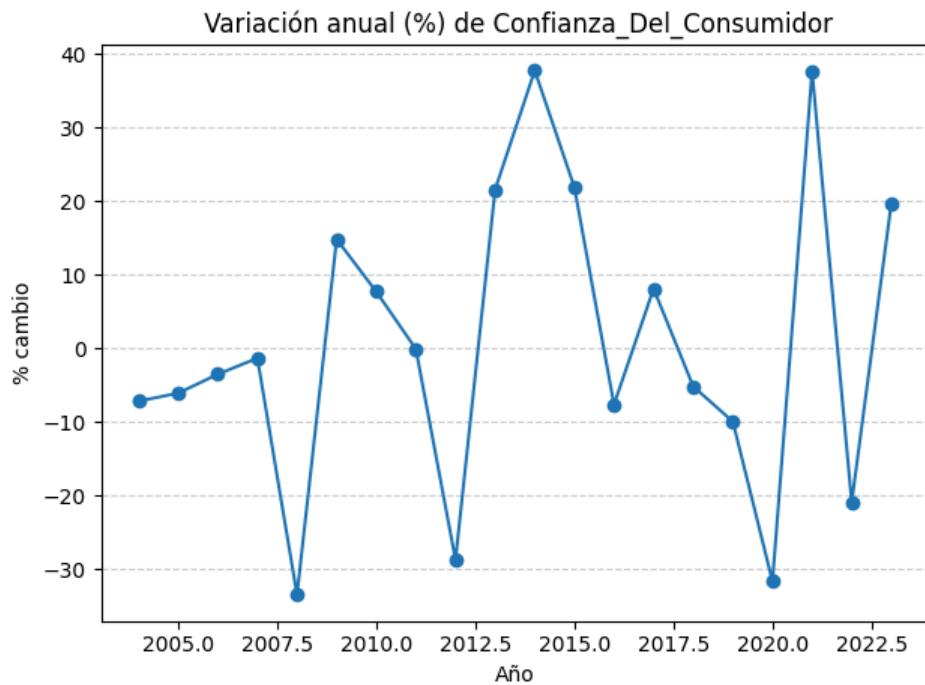


Porcentaje de variación anual para 'Confianza_Del_Consumidor':

Variación % Confianza_Del_Consumidor

Año	
2003	NaN
2004	-7.19
2005	-6.10
2006	-3.53
2007	-1.34
2008	-33.50
2009	14.82
2010	7.70
2011	-0.10
2012	-28.75
2013	21.57
2014	37.75
2015	21.90
2016	-7.65
2017	8.10
2018	-5.19
2019	-9.96
2020	-31.68
2021	37.64
2022	-20.93
2023	19.72

<Figure size 1200x400 with 0 Axes>



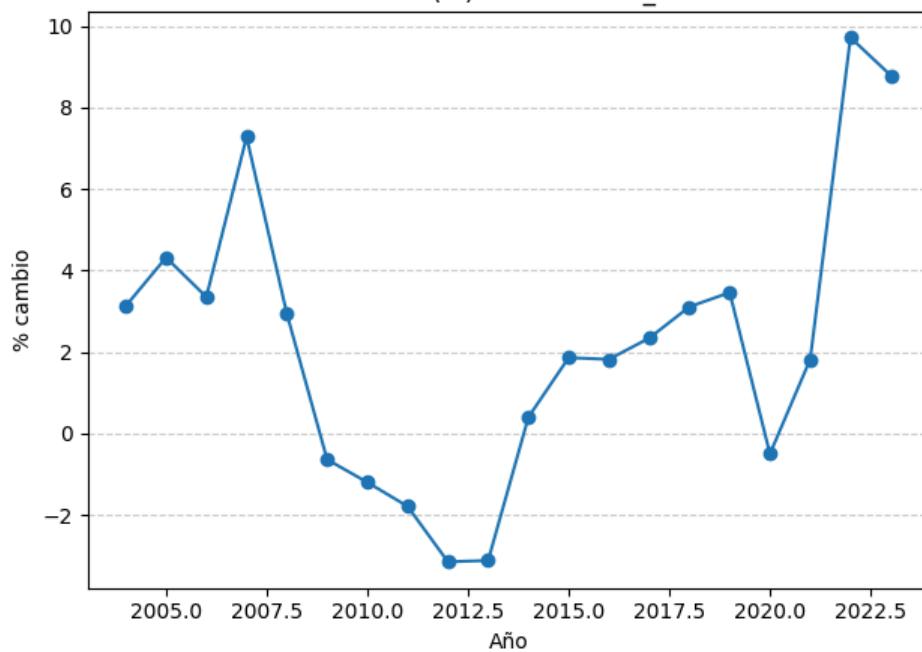
Porcentaje de variación anual para 'Contratos_Indefinidos':

Variación % Contratos_Indefinidos

Año	
2003	NaN
2004	3.13
2005	4.32
2006	3.35
2007	7.28
2008	2.94
2009	-0.64
2010	-1.20
2011	-1.79
2012	-3.15
2013	-3.12
2014	0.40
2015	1.86
2016	1.82
2017	2.34
2018	3.11
2019	3.46
2020	-0.50
2021	1.82
2022	9.72
2023	8.79

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Contratos_Indefinidos



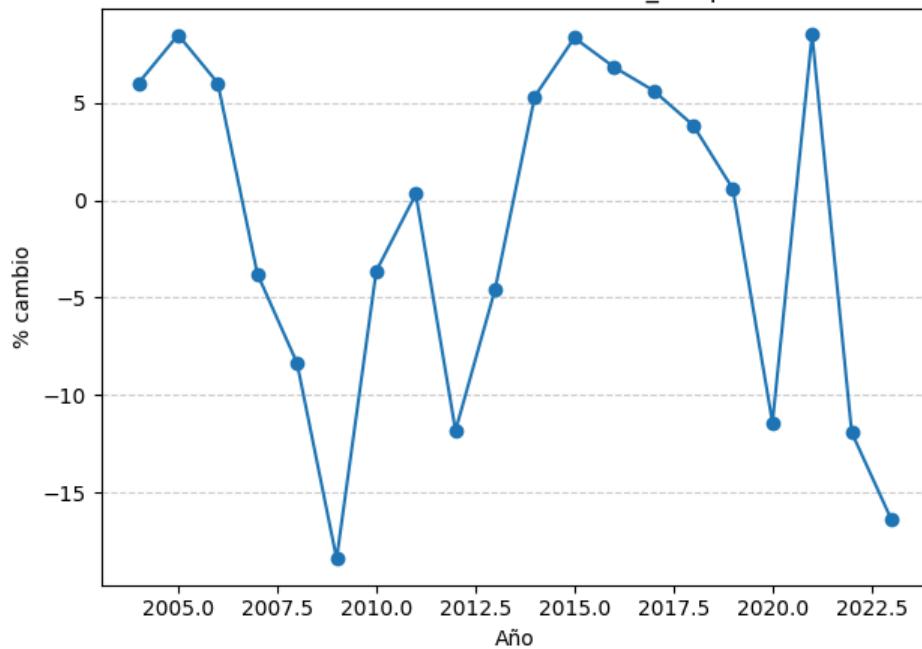
Porcentaje de variación anual para 'Contratos_Temporales':

Variación % Contratos_Temporales

Año	
2003	NaN
2004	6.01
2005	8.48
2006	6.01
2007	-3.79
2008	-8.36
2009	-18.41
2010	-3.64
2011	0.31
2012	-11.83
2013	-4.57
2014	5.32
2015	8.33
2016	6.83
2017	5.62
2018	3.84
2019	0.58
2020	-11.42
2021	8.50
2022	-11.93
2023	-16.39

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Contratos_Temporales

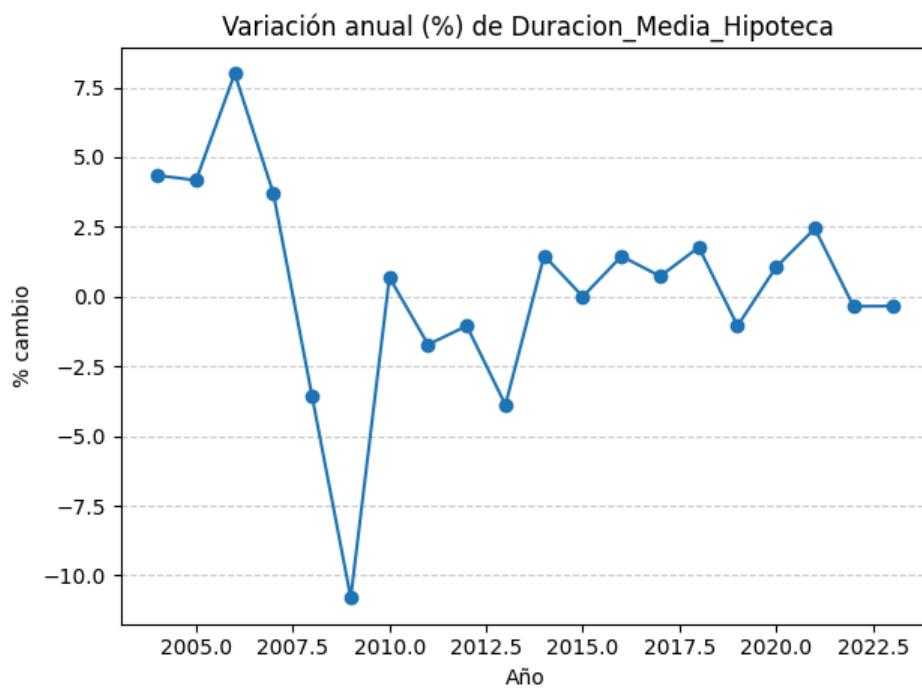


Porcentaje de variación anual para 'Duracion_Media_Hipoteca':

Variación % Duracion_Media_Hipoteca

Año	Variación %
2003	NaN
2004	4.35
2005	4.17
2006	8.00
2007	3.70
2008	-3.57
2009	-10.80
2010	0.69
2011	-1.72
2012	-1.05
2013	-3.88
2014	1.47
2015	0.00
2016	1.45
2017	0.73
2018	1.77
2019	-1.05
2020	1.06
2021	2.44
2022	-0.35
2023	-0.34

<Figure size 1200x400 with 0 Axes>

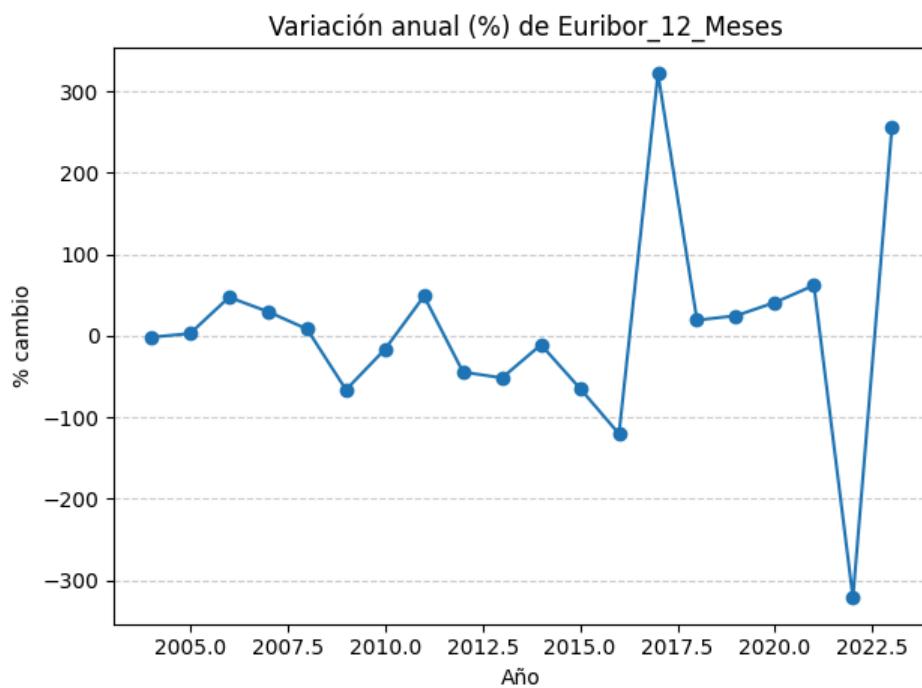


Porcentaje de variación anual para 'Euribor_12_Meses':

Variación % Euribor_12_Meses

Año	
2003	NaN
2004	-1.83
2005	2.66
2006	47.22
2007	29.49
2008	8.18
2009	-66.39
2010	-16.54
2011	48.56
2012	-44.61
2013	-51.75
2014	-11.11
2015	-64.66
2016	-120.44
2017	321.79
2018	19.12
2019	24.53
2020	40.87
2021	61.79
2022	-321.63
2023	255.32

<Figure size 1200x400 with 0 Axes>

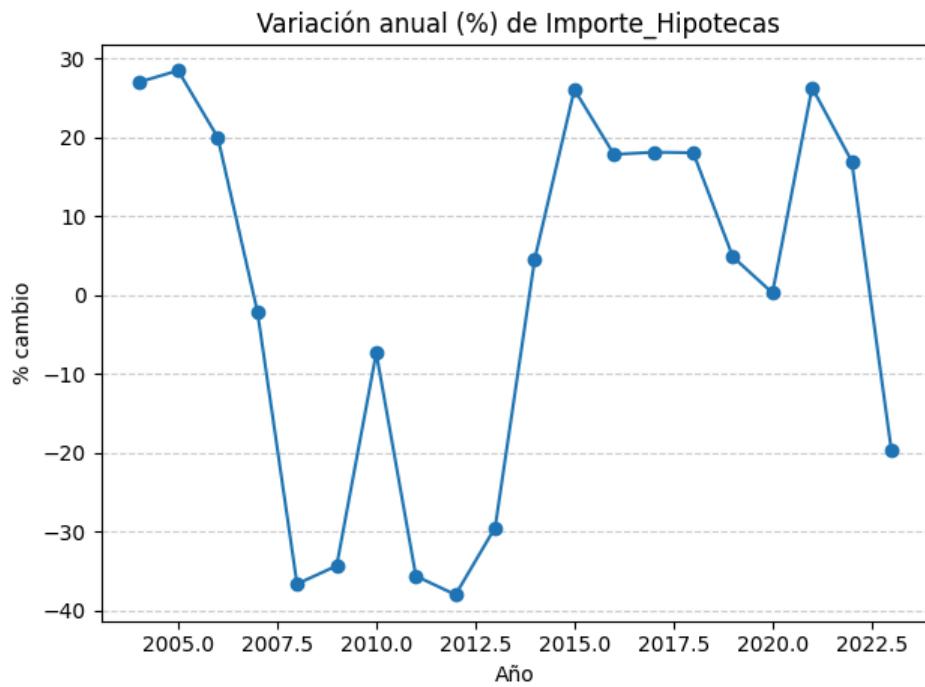


Porcentaje de variación anual para 'Importe_Hipotecas':

Variación % Importe_Hipotecas

Año	
2003	NaN
2004	27.00
2005	28.49
2006	20.00
2007	-2.08
2008	-36.66
2009	-34.36
2010	-7.35
2011	-35.65
2012	-38.03
2013	-29.50
2014	4.60
2015	26.08
2016	17.84
2017	18.13
2018	18.06
2019	4.85
2020	0.27
2021	26.33
2022	16.98
2023	-19.66

<Figure size 1200x400 with 0 Axes>

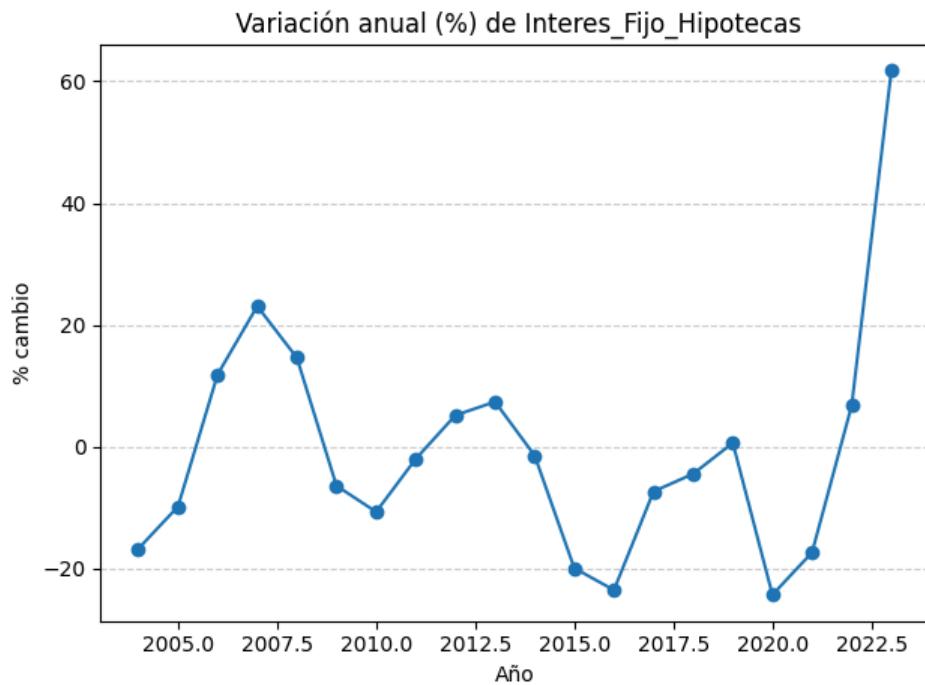


Porcentaje de variación anual para 'Interes_Fijo_Hipotecas':

Variación % Interes_Fijo_Hipotecas

Año	
2003	NaN
2004	-16.70
2005	-9.79
2006	11.89
2007	23.09
2008	14.63
2009	-6.38
2010	-10.66
2011	-1.96
2012	5.19
2013	7.40
2014	-1.41
2015	-19.89
2016	-23.49
2017	-7.31
2018	-4.42
2019	0.66
2020	-24.26
2021	-17.32
2022	6.81
2023	61.76

<Figure size 1200x400 with 0 Axes>

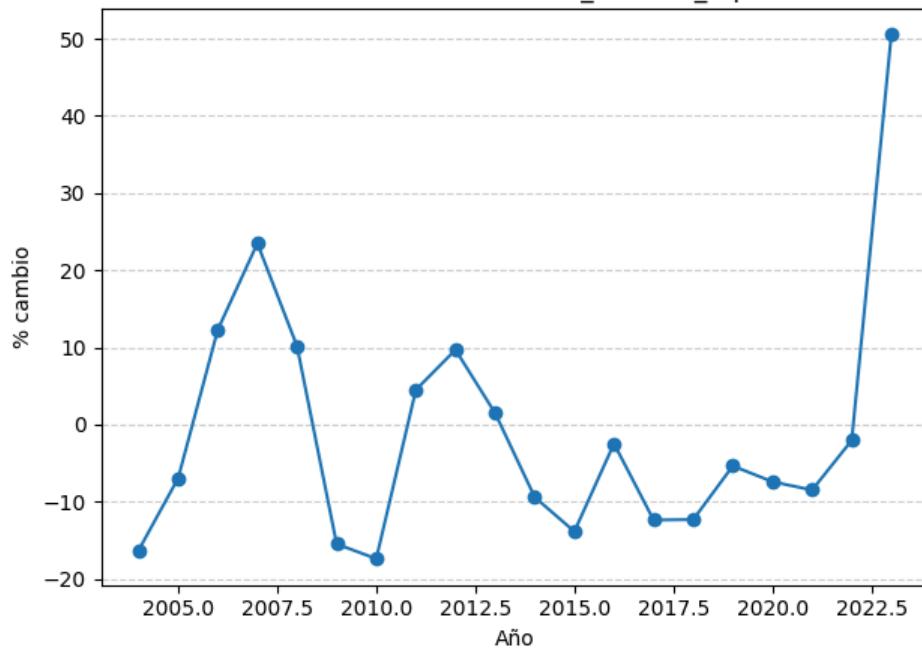


Porcentaje de variación anual para 'Interes_Variable_Hipotecas':
Variación % Interes_Variable_Hipotecas

Año	
2003	NaN
2004	-16.32
2005	-6.96
2006	12.28
2007	23.47
2008	10.15
2009	-15.49
2010	-17.40
2011	4.49
2012	9.68
2013	1.47
2014	-9.42
2015	-13.87
2016	-2.48
2017	-12.38
2018	-12.32
2019	-5.37
2020	-7.42
2021	-8.49
2022	-2.06
2023	50.53

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Interes_Variable_Hipotecas

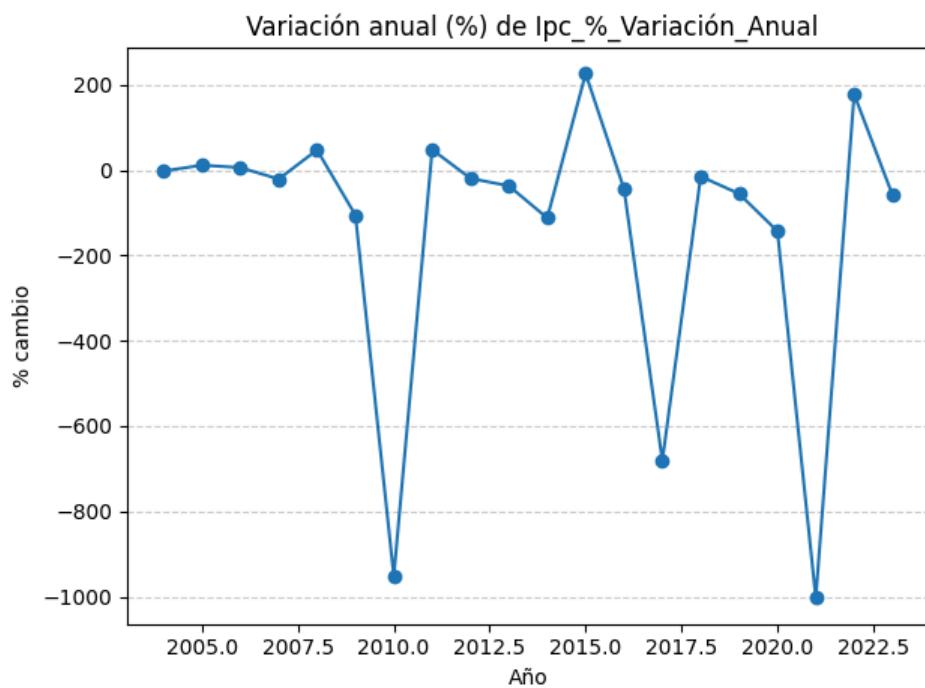


Porcentaje de variación anual para 'Ipc_%_Variación_Anual':

Variación % Ipc_%_Variación_Anual

Año	
2003	NaN
2004	-2.41
2005	11.23
2006	5.67
2007	-21.21
2008	47.04
2009	-105.84
2010	-951.72
2011	47.77
2012	-19.73
2013	-37.20
2014	-112.50
2015	226.09
2016	-44.00
2017	-680.95
2018	-15.16
2019	-55.56
2020	-143.48
2021	-1002.50
2022	177.29
2023	-58.84

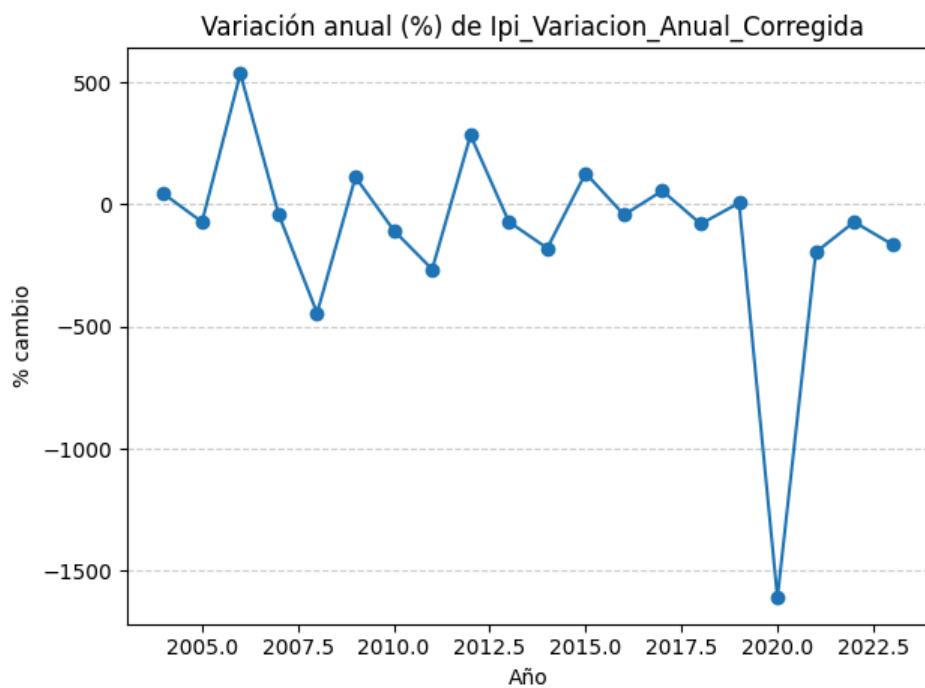
<Figure size 1200x400 with 0 Axes>



Porcentaje de variación anual para 'Ipi_Variacion_Annual_Corregida':
Variación % Ipi_Variacion_Annual_Corregida

Año	
2003	NaN
2004	43.53
2005	-72.13
2006	535.29
2007	-40.97
2008	-444.71
2009	111.49
2010	-106.46
2011	-265.83
2012	282.91
2013	-71.52
2014	-179.26
2015	127.91
2016	-43.11
2017	55.61
2018	-80.12
2019	5.80
2020	-1609.59
2021	-195.83
2022	-72.06
2023	-164.41

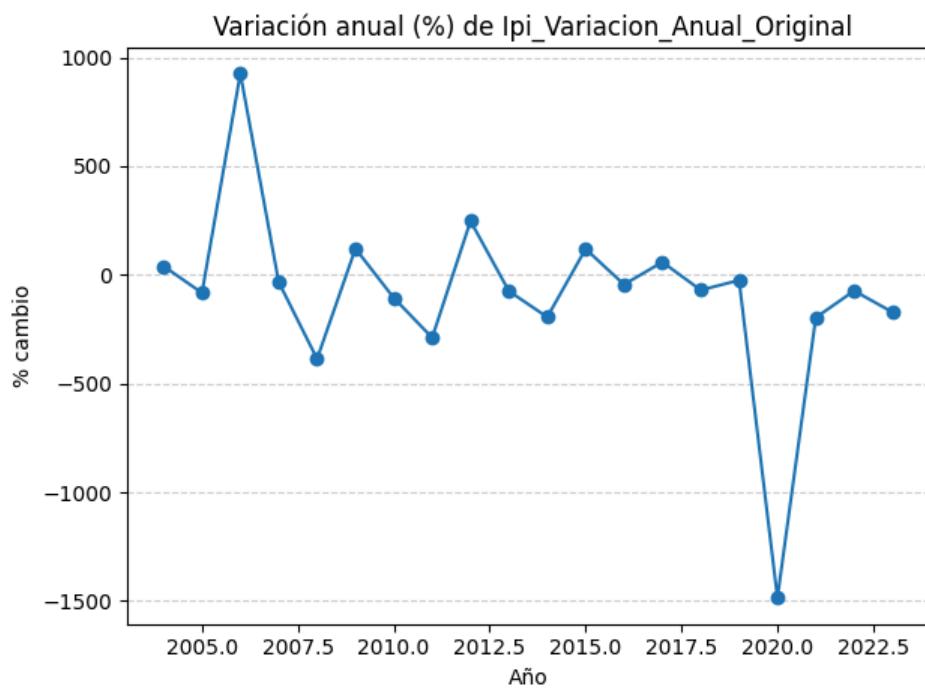
<Figure size 1200x400 with 0 Axes>



Porcentaje de variación anual para 'Ipi_Variacion_Anual_Original':
Variación % Ipi_Variacion_Anual_Original

Año	
2003	NaN
2004	39.78
2005	-83.00
2006	925.58
2007	-33.33
2008	-386.39
2009	120.67
2010	-106.19
2011	-288.70
2012	247.47
2013	-74.80
2014	-194.21
2015	118.99
2016	-44.39
2017	58.72
2018	-69.94
2019	-25.00
2020	-1484.62
2021	-196.30
2022	-73.37
2023	-171.84

<Figure size 1200x400 with 0 Axes>

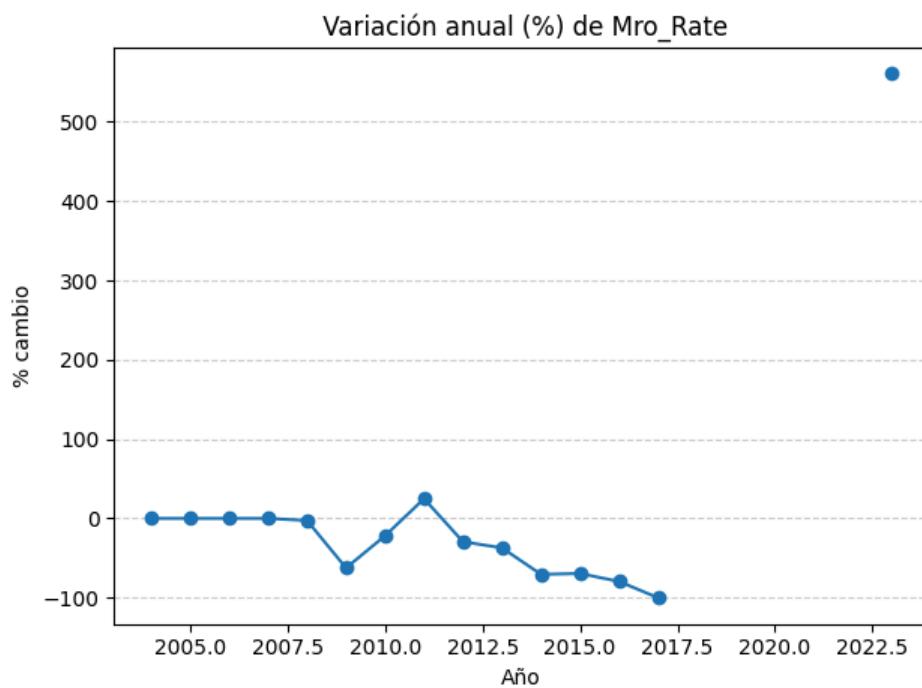


Porcentaje de variación anual para 'Mro_Rate':

Variación % Mro_Rate

Año	Variación % Mro_Rate
2003	NaN
2004	0.00
2005	0.00
2006	0.00
2007	0.00
2008	-2.62
2009	-62.42
2010	-21.92
2011	24.85
2012	-29.39
2013	-37.13
2014	-70.49
2015	-69.43
2016	-79.45
2017	-100.00
2018	NaN
2019	NaN
2020	NaN
2021	NaN
2022	-100.00
2023	560.80

<Figure size 1200x400 with 0 Axes>



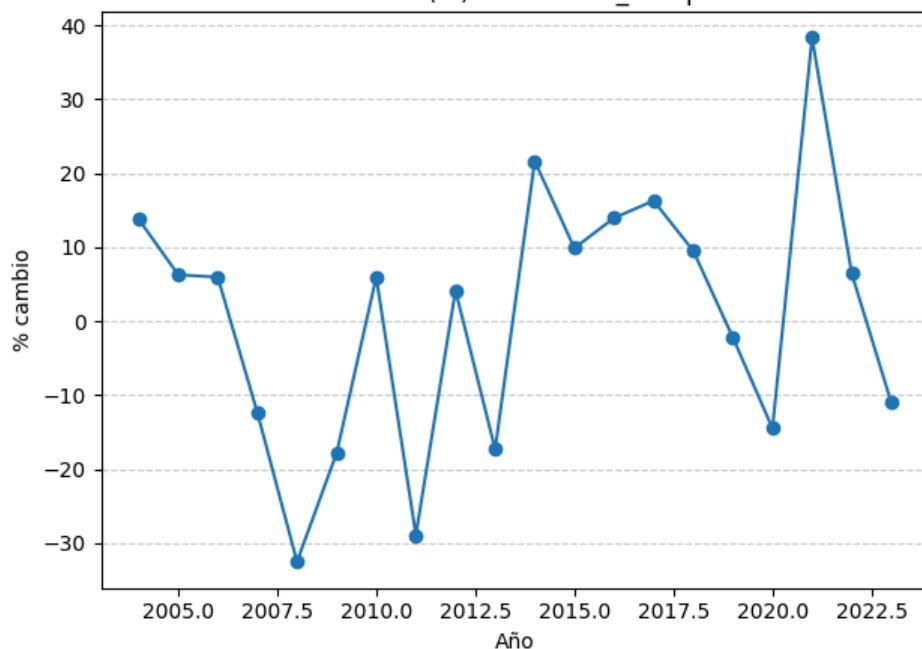
Porcentaje de variación anual para 'Número_Compraventas':

Variación % Número_Compraventas

Año	Variación % Número_Compraventas
2003	NaN
2004	13.82
2005	6.27
2006	5.95
2007	-12.39
2008	-32.55
2009	-17.85
2010	5.95
2011	-28.94
2012	4.16
2013	-17.34
2014	21.64
2015	9.87
2016	13.95
2017	16.28
2018	9.52
2019	-2.21
2020	-14.50
2021	38.36
2022	6.44
2023	-11.03

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Numero_Compraventas

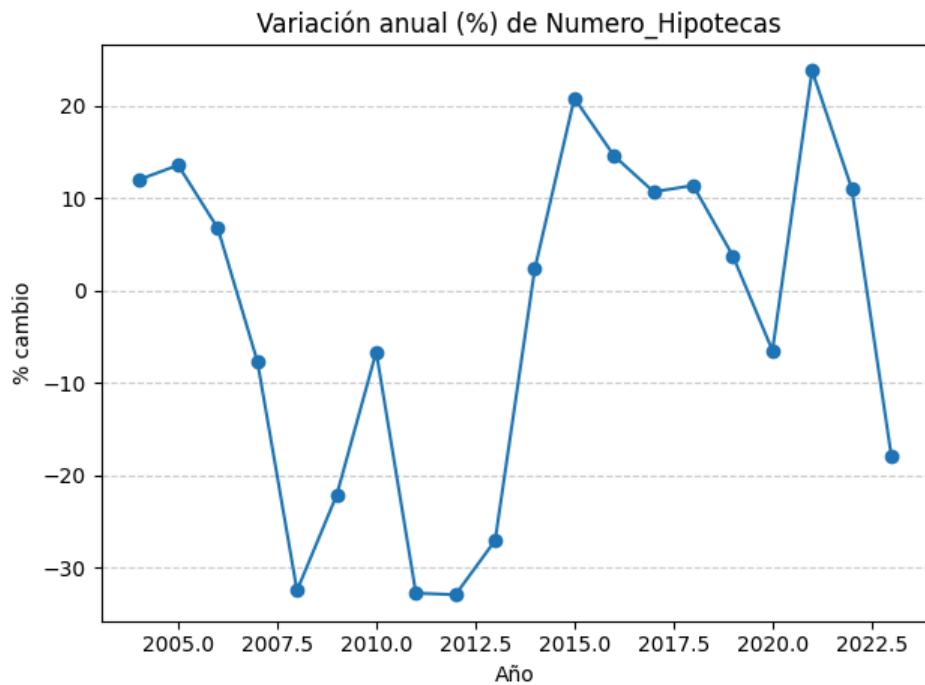


Porcentaje de variación anual para 'Numero_Hipotecas':

Variación % Numero_Hipotecas

Año	
2003	NaN
2004	11.95
2005	13.54
2006	6.72
2007	-7.70
2008	-32.49
2009	-22.18
2010	-6.66
2011	-32.77
2012	-32.95
2013	-27.08
2014	2.30
2015	20.79
2016	14.56
2017	10.66
2018	11.34
2019	3.72
2020	-6.52
2021	23.78
2022	11.01
2023	-17.91

<Figure size 1200x400 with 0 Axes>

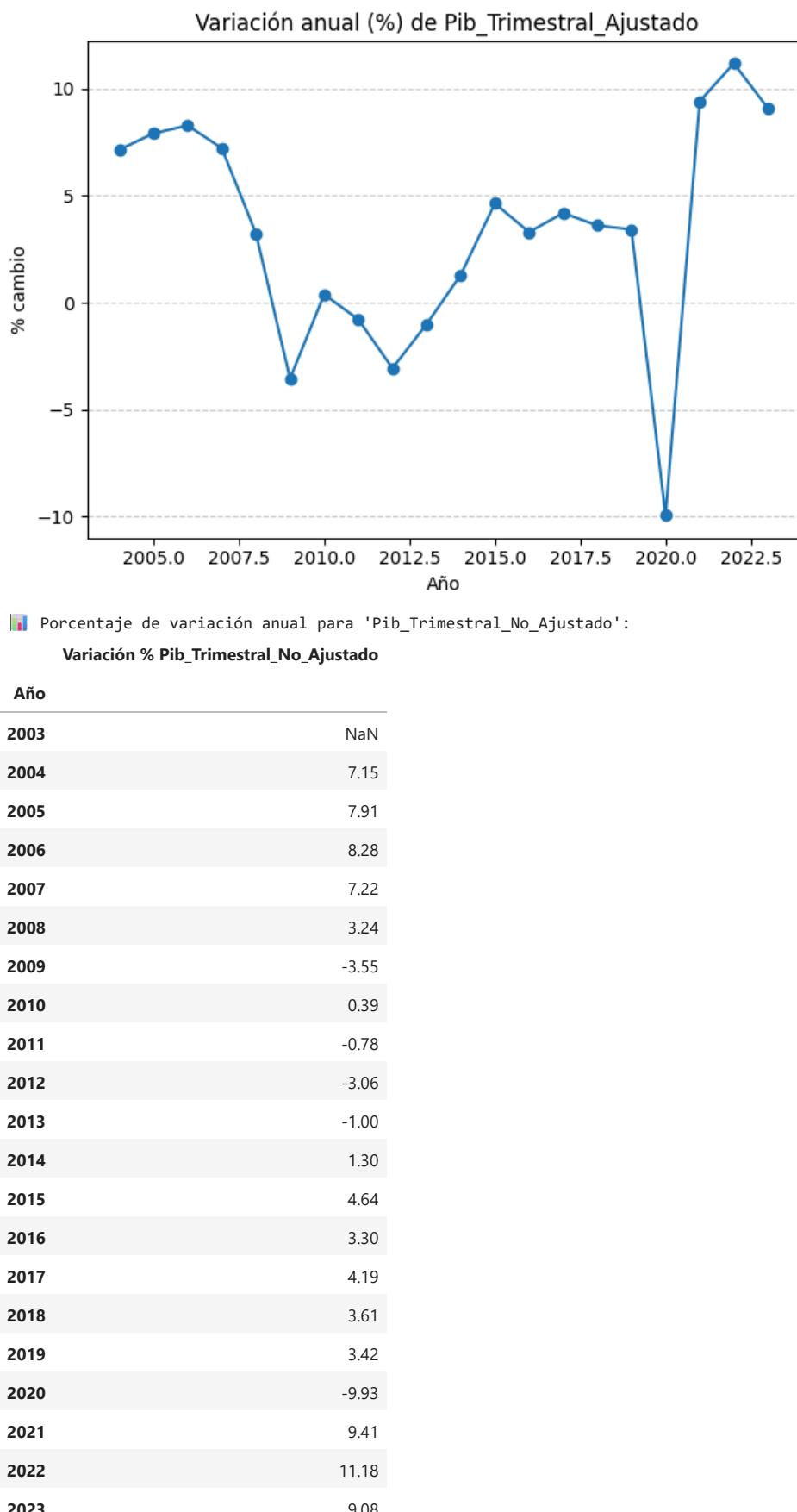


Porcentaje de variación anual para 'Pib_Trimestral_Ajustado':

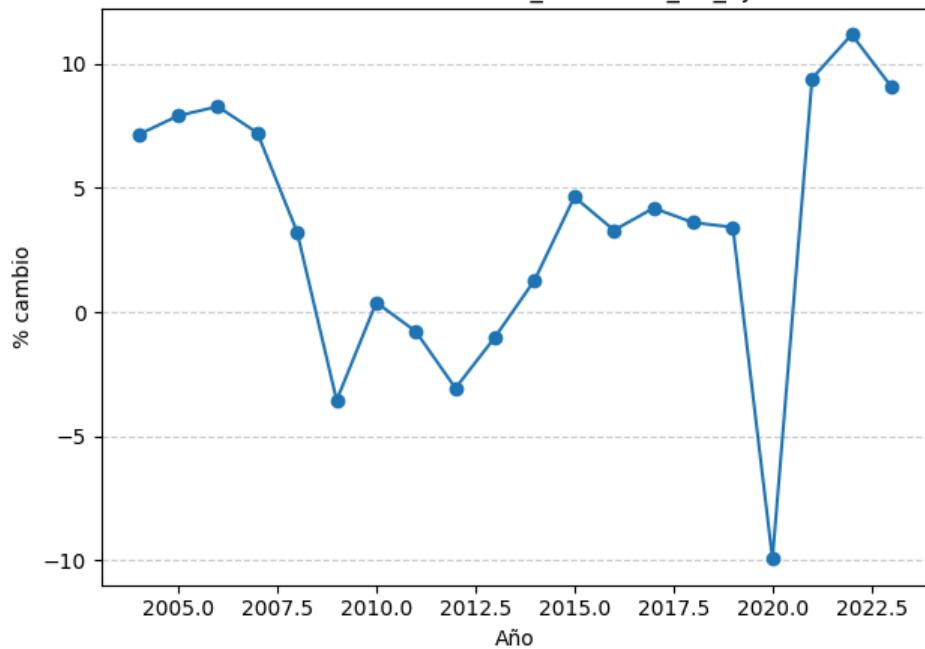
Variación % Pib_Trimestral_Ajustado

Año	
2003	NaN
2004	7.15
2005	7.91
2006	8.28
2007	7.22
2008	3.24
2009	-3.54
2010	0.39
2011	-0.78
2012	-3.06
2013	-1.00
2014	1.30
2015	4.64
2016	3.30
2017	4.19
2018	3.61
2019	3.42
2020	-9.93
2021	9.41
2022	11.18
2023	9.08

<Figure size 1200x400 with 0 Axes>



Variación anual (%) de Pib_Trimestral_No_Ajustado

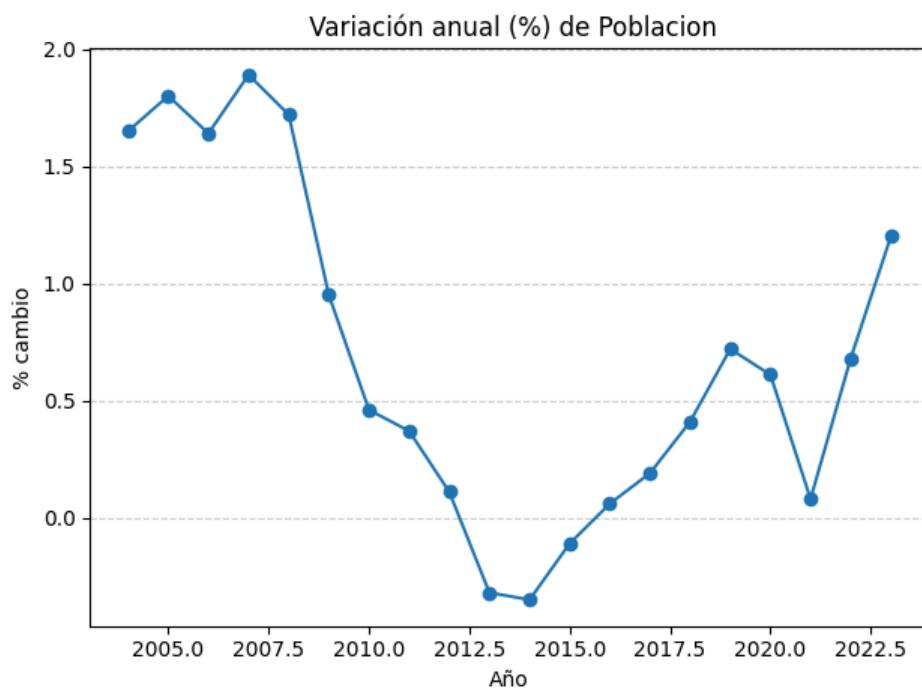


Porcentaje de variación anual para 'Poblacion':

Variación % Poblacion

Año	Variación % Poblacion
2003	NaN
2004	1.65
2005	1.80
2006	1.64
2007	1.89
2008	1.72
2009	0.95
2010	0.46
2011	0.37
2012	0.11
2013	-0.32
2014	-0.35
2015	-0.11
2016	0.06
2017	0.19
2018	0.41
2019	0.72
2020	0.61
2021	0.08
2022	0.68
2023	1.20

<Figure size 1200x400 with 0 Axes>

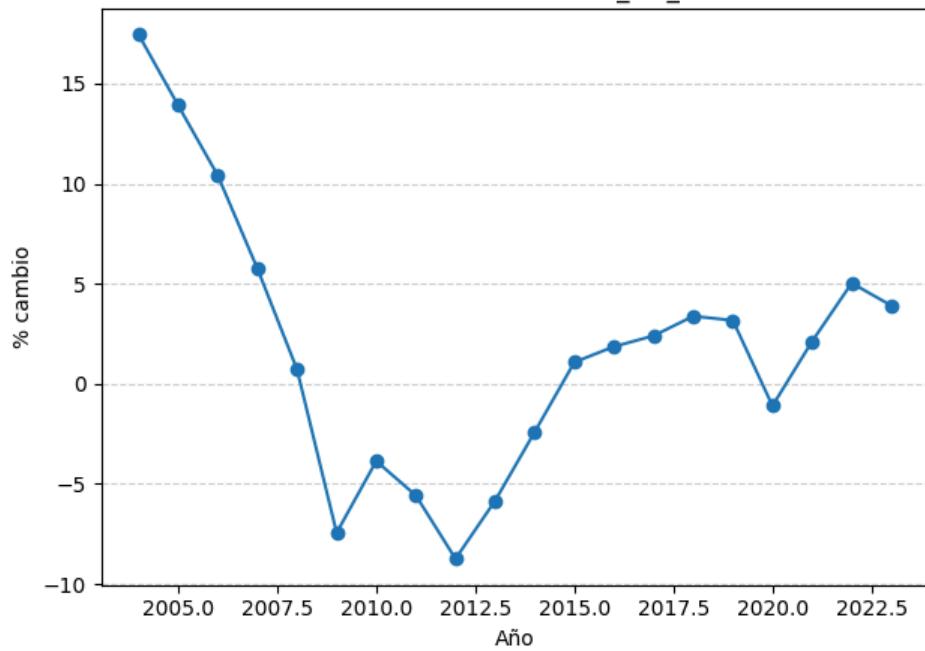


Porcentaje de variación anual para 'Precio_M2_Vivienda':
Variación % Precio_M2_Vivienda

Año	
2003	NaN
2004	17.45
2005	13.91
2006	10.41
2007	5.76
2008	0.72
2009	-7.44
2010	-3.86
2011	-5.58
2012	-8.74
2013	-5.84
2014	-2.40
2015	1.08
2016	1.86
2017	2.40
2018	3.38
2019	3.17
2020	-1.10
2021	2.12
2022	5.03
2023	3.91

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Precio_M2_Vivienda



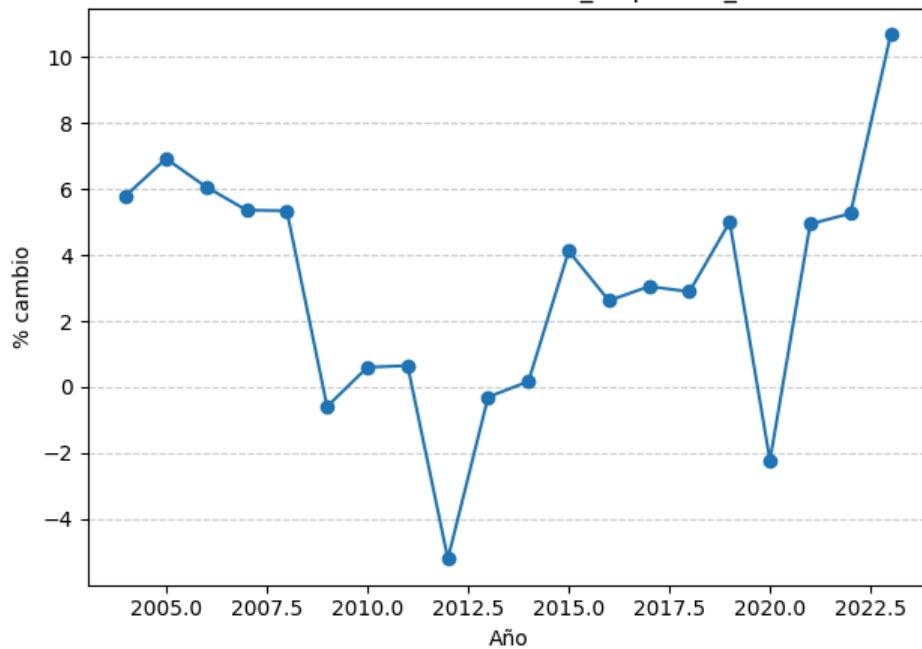
Porcentaje de variación anual para 'Renta_Disponible_Bruta':

Variación % Renta_Disponible_Bruta

Año	
2003	NaN
2004	5.80
2005	6.92
2006	6.06
2007	5.36
2008	5.34
2009	-0.59
2010	0.60
2011	0.65
2012	-5.20
2013	-0.30
2014	0.17
2015	4.12
2016	2.62
2017	3.05
2018	2.89
2019	5.00
2020	-2.22
2021	4.94
2022	5.26
2023	10.68

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Renta_Disponible_Bruta

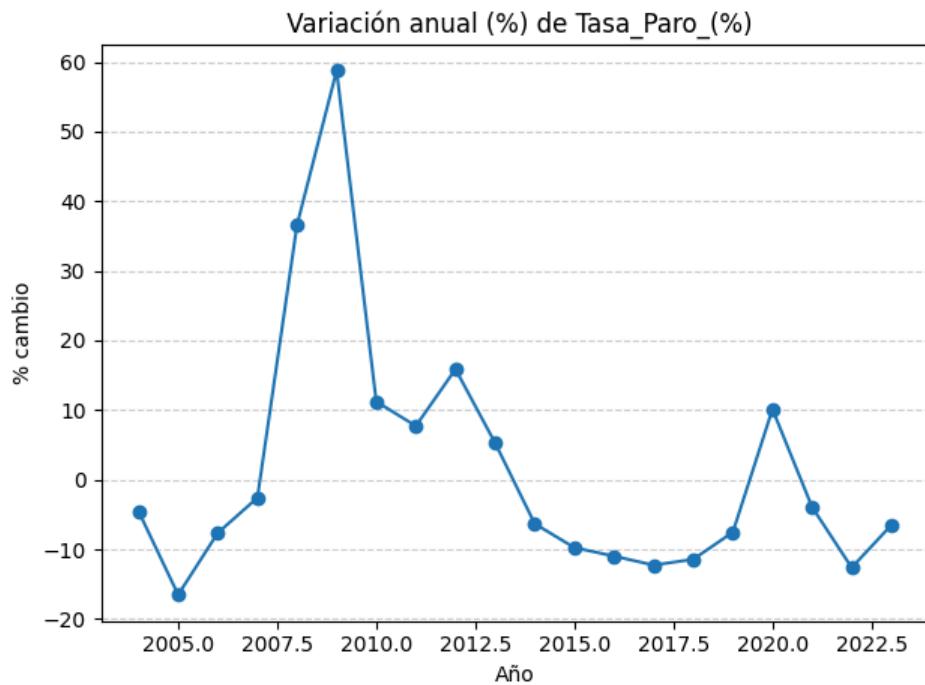


Porcentaje de variación anual para 'Tasa_Paro_(%)':

Variación % Tasa_Paro_(%)

Año	
2003	NaN
2004	-4.53
2005	-16.53
2006	-7.65
2007	-2.60
2008	36.59
2009	58.78
2010	11.22
2011	7.72
2012	15.88
2013	5.27
2014	-6.33
2015	-9.76
2016	-10.98
2017	-12.27
2018	-11.44
2019	-7.54
2020	10.12
2021	-3.94
2022	-12.63
2023	-6.52

<Figure size 1200x400 with 0 Axes>



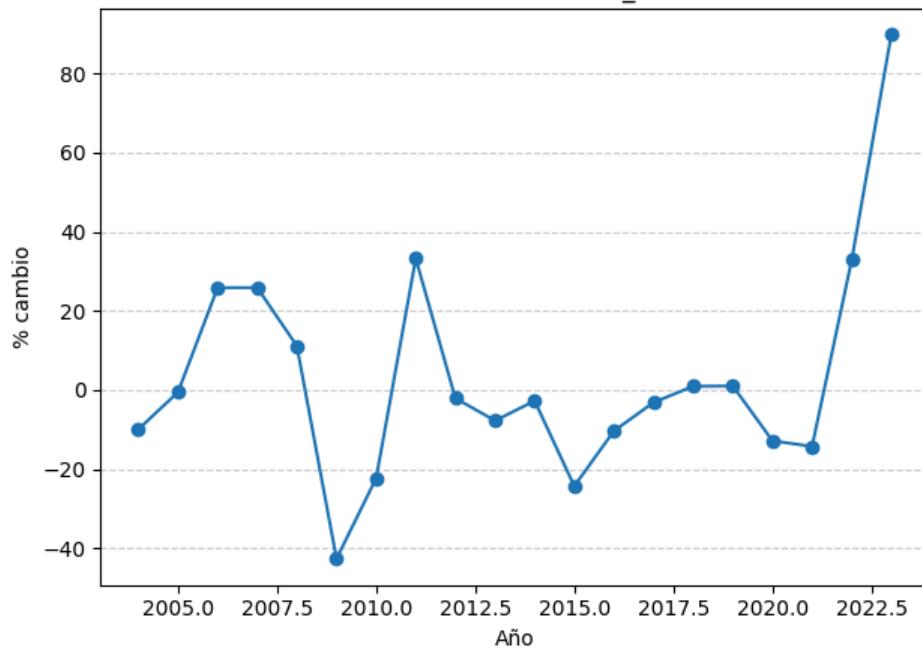
Porcentaje de variación anual para 'Ti_Credito':

Variación % Ti_Credito

Año	Variación % Ti_Credito
2003	NaN
2004	-9.92
2005	-0.53
2006	25.85
2007	25.86
2008	11.15
2009	-42.61
2010	-22.29
2011	33.22
2012	-2.07
2013	-7.73
2014	-2.73
2015	-24.27
2016	-10.33
2017	-3.12
2018	0.95
2019	1.04
2020	-12.80
2021	-14.25
2022	32.84
2023	89.80

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Ti_Credito

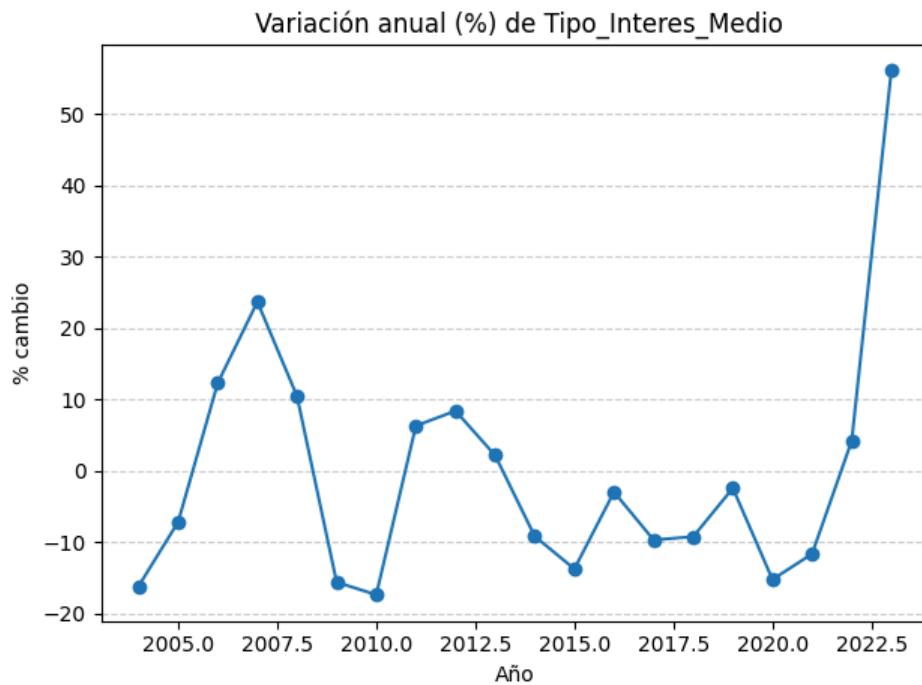


Porcentaje de variación anual para 'Tipo_Interes_Medio':

Variación % Tipo_Interes_Medio

Año	
2003	NaN
2004	-16.18
2005	-7.17
2006	12.30
2007	23.67
2008	10.45
2009	-15.61
2010	-17.37
2011	6.30
2012	8.39
2013	2.16
2014	-9.18
2015	-13.75
2016	-2.97
2017	-9.69
2018	-9.23
2019	-2.44
2020	-15.23
2021	-11.64
2022	4.20
2023	56.07

<Figure size 1200x400 with 0 Axes>

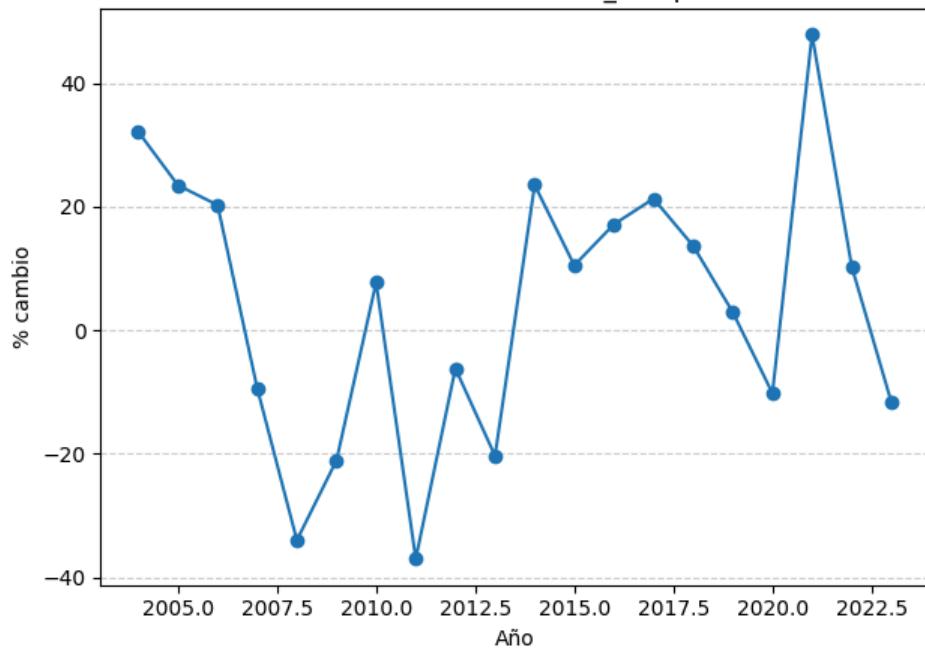


Porcentaje de variación anual para 'Valor_Compraventa':
Variación % Valor_Compraventa

Año	
2003	NaN
2004	32.10
2005	23.46
2006	20.25
2007	-9.39
2008	-33.96
2009	-21.04
2010	7.74
2011	-36.96
2012	-6.17
2013	-20.31
2014	23.50
2015	10.50
2016	17.14
2017	21.29
2018	13.66
2019	2.91
2020	-10.18
2021	47.84
2022	10.32
2023	-11.61

<Figure size 1200x400 with 0 Axes>

Variación anual (%) de Valor_Compraventa

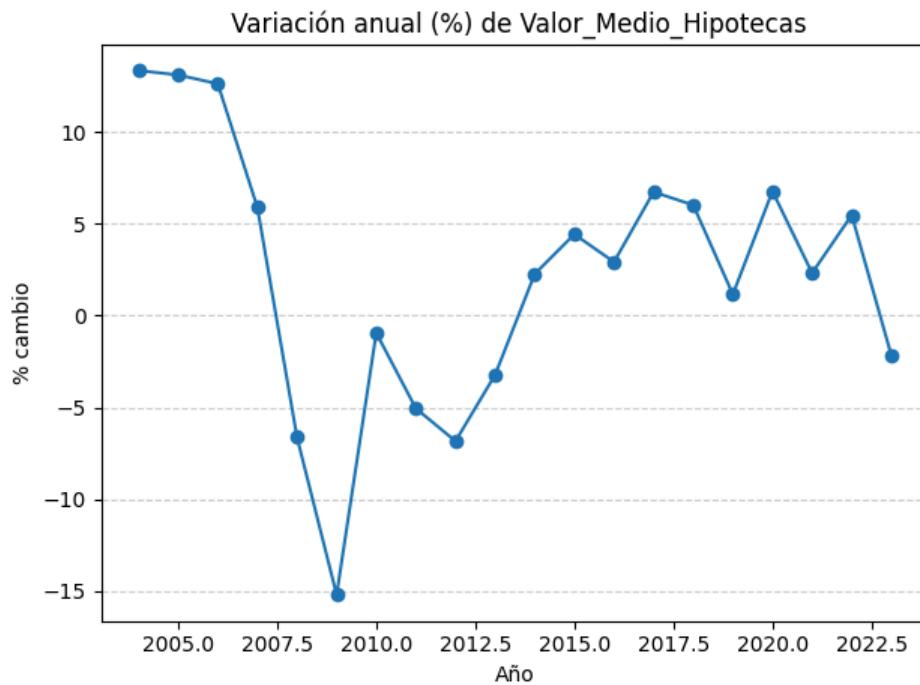


Porcentaje de variación anual para 'Valor_Medio_Hipotecas':

Variación % Valor_Medio_Hipotecas

Año	
2003	NaN
2004	13.34
2005	13.10
2006	12.62
2007	5.89
2008	-6.62
2009	-15.19
2010	-0.94
2011	-5.05
2012	-6.83
2013	-3.22
2014	2.27
2015	4.43
2016	2.92
2017	6.73
2018	6.02
2019	1.16
2020	6.77
2021	2.32
2022	5.47
2023	-2.16

<Figure size 1200x400 with 0 Axes>



Variación trimestral de las variables

De igual forma, vamos a observar también para cada variable:

- **Variación trimestral secuencial:** para ver cómo cambian de un trimestre al siguiente
- **Variación interanual por trimestre:** por ejemplo, de Q1 de un año a Q1 del siguiente
- **Gráfico para ver cómo evoluciona cada trimestre (Q1, Q2, Q3, Q4) a lo largo de los años para cada variable.**: esto es útil para detectar cambios estructurales dentro de un mismo trimestre con el paso del tiempo, ver si hay trimestres que ganan o pierden importancia con el tiempo e identificar cambios de tendencia por estación

```
In [17]: df_clean['Periodo'] = df_clean['Fecha'].dt.to_period('Q')

df_clean = df_clean.sort_values('Periodo')

print("⌚ Variación % de un trimestre al siguiente (secuencial):")

for col in cols:
    trimestral_mean = df_clean.groupby('Periodo')[col].mean()
    var_trimestral = trimestral_mean.pct_change() * 100
    var_trimestral_df = var_trimestral.to_frame(name=f'Variación % trimestral de {col}').round(2)

    print(f"\n📊 {col} - variación secuencial por trimestre:")
    display(var_trimestral_df)
```

⌚ Variación % de un trimestre al siguiente (secuencial):

📊 Afiliaciones_Ss - variación secuencial por trimestre:

Variación % trimestral de Afiliaciones_Ss

Periodo	
2003Q1	NaN
2003Q2	2.03
2003Q3	0.31
2003Q4	0.07
2004Q1	0.50
...	...
2022Q4	0.58
2023Q1	-0.06
2023Q2	2.46
2023Q3	-0.11
2023Q4	0.40

84 rows × 1 columns

 Cantidad_De_Extranjeros - variación secuencial por trimestre:**Variación % trimestral de Cantidad_De_Extranjeros**

Periodo	
2003Q1	NaN
2003Q2	6.01
2003Q3	5.73
2003Q4	4.81
2004Q1	4.54
...	...
2022Q4	2.66
2023Q1	2.50
2023Q2	2.01
2023Q3	1.19
2023Q4	1.83

84 rows × 1 columns

 Confianza_Del_Consumidor - variación secuencial por trimestre:**Variación % trimestral de Confianza_Del_Consumidor**

Periodo	
2003Q1	NaN
2003Q2	-1.58
2003Q3	-1.58
2003Q4	-1.58
2004Q1	-1.58
...	...
2022Q4	3.60
2023Q1	20.66
2023Q2	18.37
2023Q3	4.29
2023Q4	-12.67

84 rows × 1 columns

📊 Contratos_Indefinidos - variación secuencial por trimestre:

Variación % trimestral de Contratos_Indefinidos

Periodo

2003Q1	NaN
2003Q2	1.39
2003Q3	1.19
2003Q4	0.43
2004Q1	0.57
...	...
2022Q4	2.45
2023Q1	0.68
2023Q2	3.03
2023Q3	1.53
2023Q4	0.30

84 rows × 1 columns

📊 Contratos_Temporales - variación secuencial por trimestre:

Variación % trimestral de Contratos_Temporales

Periodo

2003Q1	NaN
2003Q2	4.13
2003Q3	2.23
2003Q4	0.76
2004Q1	-1.43
...	...
2022Q4	-11.34
2023Q1	-3.85
2023Q2	3.01
2023Q3	0.74
2023Q4	-5.10

84 rows × 1 columns

📊 Duracion_Media_Hipoteca - variación secuencial por trimestre:

Variación % trimestral de Duracion_Media_Hipoteca

Periodo	
2003Q1	NaN
2003Q2	0.00
2003Q3	0.00
2003Q4	0.00
2004Q1	4.35
...	...
2022Q4	1.40
2023Q1	-1.38
2023Q2	1.40
2023Q3	-1.38
2023Q4	-1.36

84 rows × 1 columns

| Euribor_12_Meses - variación secuencial por trimestre:

Variación % trimestral de Euribor_12_Meses

Periodo	
2003Q1	NaN
2003Q2	-8.95
2003Q3	-1.49
2003Q4	7.27
2004Q1	-9.30
...	...
2022Q4	89.43
2023Q1	24.11
2023Q2	10.53
2023Q3	6.41
2023Q4	-4.12

84 rows × 1 columns

| Importe_Hipotecas - variación secuencial por trimestre:

Variación % trimestral de Importe_Hipotecas

Periodo	
2003Q1	NaN
2003Q2	-3.21
2003Q3	2.47
2003Q4	2.68
2004Q1	21.94
...	...
2022Q4	-3.61
2023Q1	-4.77
2023Q2	-14.96
2023Q3	-5.17
2023Q4	1.66

84 rows × 1 columns

📊 Interes_Fijo_Hipotecas - variación secuencial por trimestre:

Variación % trimestral de Interes_Fijo_Hipotecas

Periodo

2003Q1	NaN
2003Q2	0.00
2003Q3	0.00
2003Q4	0.00
2004Q1	-16.70
...	...
2022Q4	0.00
2023Q1	61.76
2023Q2	0.00
2023Q3	0.00
2023Q4	0.00

84 rows × 1 columns

📊 Interes_Variable_Hipotecas - variación secuencial por trimestre:

Variación % trimestral de Interes_Variable_Hipotecas

Periodo

2003Q1	NaN
2003Q2	0.00
2003Q3	0.00
2003Q4	0.00
2004Q1	-16.32
...	...
2022Q4	0.00
2023Q1	50.53
2023Q2	0.00
2023Q3	0.00
2023Q4	0.00

84 rows × 1 columns

📊 Ipc_%_Variación_Anual - variación secuencial por trimestre:

Variación % trimestral de Ipc_%_Variación_Annal**Periodo**

2003Q1	NaN
2003Q2	-23.68
2003Q3	3.45
2003Q4	-7.78
2004Q1	-19.28
...	...
2022Q4	-35.43
2023Q1	-23.08
2023Q2	-44.67
2023Q3	-6.02
2023Q4	29.49

84 rows × 1 columns

 Ipi_Variacion_Annal_Corregida - variación secuencial por trimestre:**Variación % trimestral de Ipi_Variacion_Annal_Corregida****Periodo**

2003Q1	NaN
2003Q2	13.95
2003Q3	-46.94
2003Q4	100.00
2004Q1	7.69
...	...
2022Q4	-105.17
2023Q1	-300.00
2023Q2	-816.67
2023Q3	-4.65
2023Q4	-58.54

84 rows × 1 columns

 Ipi_Variacion_Annal_Original - variación secuencial por trimestre:**Variación % trimestral de Ipi_Variacion_Annal_Original****Periodo**

2003Q1	NaN
2003Q2	-114.66
2003Q3	-252.94
2003Q4	115.38
2004Q1	10.71
...	...
2022Q4	-128.85
2023Q1	-250.00
2023Q2	-293.33
2023Q3	34.48
2023Q4	-65.81

84 rows × 1 columns

📊 Mro_Rate - variación secuencial por trimestre:

Variación % trimestral de Mro_Rate

Periodo

2003Q1	NaN
2003Q2	0.00
2003Q3	0.00
2003Q4	0.00
2004Q1	0.00
...	...
2022Q4	132.38
2023Q1	144.24
2023Q2	14.85
2023Q3	8.48
2023Q4	4.65

84 rows × 1 columns

📊 Numero_Compraventas - variación secuencial por trimestre:

Variación % trimestral de Numero_Compraventas

Periodo

2003Q1	NaN
2003Q2	0.88
2003Q3	0.88
2003Q4	0.88
2004Q1	0.88
...	...
2022Q4	3.08
2023Q1	-9.19
2023Q2	7.95
2023Q3	-16.58
2023Q4	17.90

84 rows × 1 columns

📊 Numero_Hipotecas - variación secuencial por trimestre:

Variación % trimestral de Número_Hipotecas

Periodo	
2003Q1	NaN
2003Q2	-6.27
2003Q3	-4.09
2003Q4	2.03
2004Q1	17.11
...	...
2022Q4	-4.56
2023Q1	-2.18
2023Q2	-14.06
2023Q3	-5.67
2023Q4	1.69

84 rows × 1 columns

📊 Pib_Trimestral_Ajustado - variación secuencial por trimestre:

Variación % trimestral de Pib_Trimestral_Ajustado

Periodo	
2003Q1	NaN
2003Q2	1.87
2003Q3	1.46
2003Q4	1.76
2004Q1	1.73
...	...
2022Q4	3.98
2023Q1	3.00
2023Q2	0.27
2023Q3	1.04
2023Q4	2.88

84 rows × 1 columns

📊 Pib_Trimestral_No_Ajustado - variación secuencial por trimestre:

Variación % trimestral de Pib_Trimestral_No_Ajustado

Periodo	
2003Q1	NaN
2003Q2	6.27
2003Q3	-3.11
2003Q4	7.96
2004Q1	-3.94
...	...
2022Q4	7.72
2023Q1	-3.09
2023Q2	5.37
2023Q3	-1.67
2023Q4	7.15

84 rows × 1 columns

📊 Poblacion - variación secuencial por trimestre:

Variación % trimestral de Poblacion

Periodo

2003Q1	NaN
2003Q2	0.44
2003Q3	0.44
2003Q4	0.42
2004Q1	0.41
...	...
2022Q4	0.33
2023Q1	0.30
2023Q2	0.25
2023Q3	0.24
2023Q4	0.34

84 rows × 1 columns

📊 Precio_M2_Vivienda - variación secuencial por trimestre:

Variación % trimestral de Precio_M2_Vivienda

Periodo

2003Q1	NaN
2003Q2	6.45
2003Q3	2.70
2003Q4	2.63
2004Q1	5.50
...	...
2022Q4	0.53
2023Q1	2.24
2023Q2	0.27
2023Q3	1.07
2023Q4	1.65

84 rows × 1 columns

📊 Renta_Disponible_Bruta - variación secuencial por trimestre:

Variación % trimestral de Renta_Disponible_Bruta

Periodo	
2003Q1	NaN
2003Q2	15.96
2003Q3	-9.97
2003Q4	10.44
2004Q1	-8.08
...	...
2022Q4	19.72
2023Q1	-10.55
2023Q2	21.41
2023Q3	-14.12
2023Q4	17.63

84 rows × 1 columns

📊 Tasa_Paro_(%) - variación secuencial por trimestre:

Variación % trimestral de Tasa_Paro_(%)

Periodo	
2003Q1	NaN
2003Q2	-5.92
2003Q3	0.18
2003Q4	0.62
2004Q1	1.14
...	...
2022Q4	2.04
2023Q1	3.00
2023Q2	-12.78
2023Q3	1.89
2023Q4	-0.76

84 rows × 1 columns

📊 Ti_Credito - variación secuencial por trimestre:

Variación % trimestral de Ti_Credito

Periodo	
2003Q1	NaN
2003Q2	-6.86
2003Q3	-9.27
2003Q4	-1.09
2004Q1	-0.71
...	...
2022Q4	36.02
2023Q1	26.18
2023Q2	8.71
2023Q3	3.41
2023Q4	-0.96

84 rows × 1 columns

📊 Tipo_Interes_Medio - variación secuencial por trimestre:

Variación % trimestral de Tipo_Interes_Medio

Periodo

2003Q1	NaN
2003Q2	-5.11
2003Q3	-5.39
2003Q4	-7.83
2004Q1	-3.18
...	...
2022Q4	25.38
2023Q1	15.22
2023Q2	10.85
2023Q3	3.62
2023Q4	1.64

84 rows × 1 columns

📊 Valor_Compraventa - variación secuencial por trimestre:

Variación % trimestral de Valor_Compraventa

Periodo

2003Q1	NaN
2003Q2	3.80
2003Q3	3.80
2003Q4	3.80
2004Q1	3.80
...	...
2022Q4	-0.45
2023Q1	-9.52
2023Q2	11.30
2023Q3	-15.66
2023Q4	15.91

84 rows × 1 columns

📊 Valor_Medio_Hipotecas - variación secuencial por trimestre:

Variación % trimestral de Valor_Medio_Hipotecas

Periodo	
2003Q1	NaN
2003Q2	3.27
2003Q3	6.83
2003Q4	0.64
2004Q1	4.12
...	...
2022Q4	1.00
2023Q1	-2.64
2023Q2	-1.05
2023Q3	0.54
2023Q4	-0.03

84 rows × 1 columns

```
In [18]: df_clean['Año'] = df_clean['Fecha'].dt.year
df_clean['Trimestre_simple'] = df_clean['Fecha'].dt.quarter.map({1: 'Q1', 2: 'Q2', 3: 'Q3', 4: 'Q4'})

print("\n⏳ Variación % interanual por trimestre (Q1 vs Q1, Q2 vs Q2, etc.) con trimestres como columnas:")

for col in cols:
    mean_trim = df_clean.groupby(['Año', 'Trimestre_simple'])[col].mean().unstack()

    var_interanual = mean_trim.pct_change() * 100
    var_interanual = var_interanual.round(2)

    print(f"\n📊 Variación interanual (%) de {col} por trimestre:")
    display(var_interanual)
```

⏳ Variación % interanual por trimestre (Q1 vs Q1, Q2 vs Q2, etc.) con trimestres como columnas:

📊 Variación interanual (%) de Afiliaciones_Ss por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	2.94	2.58	2.67	3.09
2005	2.80	3.93	5.29	5.58
2006	5.52	4.75	3.59	3.27
2007	3.64	3.33	2.65	2.36
2008	1.42	0.27	-0.91	-3.40
2009	-5.51	-6.35	-5.84	-3.96
2010	-2.64	-1.67	-0.98	-0.89
2011	-1.04	-0.80	-1.18	-2.00
2012	-2.49	-3.00	-3.34	-4.89
2013	-5.19	-4.63	-3.99	-1.38
2014	0.36	1.59	1.99	2.43
2015	3.24	3.37	2.80	3.27
2016	2.76	2.72	3.27	2.98
2017	3.24	3.67	3.50	3.44
2018	3.53	3.32	2.89	2.91
2019	2.91	2.77	2.80	2.50
2020	0.43	-4.14	-2.79	-1.68
2021	-0.52	3.71	3.72	4.01
2022	4.45	4.96	3.46	2.39
2023	2.62	2.70	2.88	2.69

📊 Variación interanual (%) de Cantidad_De_Extranjeros por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año

2003	NaN	NaN	NaN	NaN
2004	22.81	20.03	17.53	17.94
2005	18.25	18.79	19.30	16.81
2006	14.60	13.27	12.04	12.64
2007	13.19	14.21	15.19	14.71
2008	14.31	12.04	9.91	7.84
2009	5.91	4.50	3.13	1.68
2010	0.30	-0.55	-1.41	-1.54
2011	-1.67	-1.62	-1.58	-1.50
2012	-1.44	-1.97	-2.52	-2.90
2013	-3.27	-4.29	-5.33	-6.50
2014	-7.67	-7.35	-7.02	-5.90
2015	-4.76	-3.59	-2.37	-1.57
2016	-0.78	-0.71	-0.65	-0.34
2017	-0.04	0.71	1.48	2.56
2018	3.61	4.30	4.99	5.49
2019	5.97	6.85	7.72	7.88
2020	8.05	7.11	6.19	4.60
2021	3.08	1.76	0.44	1.14
2022	1.97	4.55	7.79	9.29
2023	10.54	10.31	8.61	7.74

📊 Variación interanual (%) de Confianza_Del_Consumidor por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	2003	NaN	NaN	NaN	NaN
2004	-6.17	-6.17	-6.17	-10.34	
2005	-4.91	-7.16	-6.51	-5.82	
2006	-6.42	-6.90	-3.94	3.68	
2007	1.16	10.20	-0.52	-15.67	
2008	-18.12	-38.75	-43.18	-34.23	
2009	-30.95	11.68	53.44	48.38	
2010	45.99	8.92	-2.07	-9.91	
2011	-4.54	6.32	-3.78	2.23	
2012	-6.80	-31.95	-42.51	-33.80	
2013	-19.93	11.86	64.79	54.43	
2014	42.27	54.34	31.83	26.45	
2015	33.04	17.22	19.44	19.46	
2016	-4.38	-8.43	-10.88	-6.75	
2017	2.13	13.35	13.12	4.04	
2018	2.60	-6.76	-6.59	-9.60	
2019	-5.59	0.81	-19.06	-16.47	
2020	-16.77	-45.28	-37.01	-26.75	
2021	-22.78	60.83	84.79	57.32	
2022	27.69	-19.91	-40.81	-34.35	
2023	-10.48	17.19	54.32	30.09	

📊 Variación interanual (%) de Contratos_Indefinidos por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	3.63	3.14	2.30	3.45
2005	4.47	4.52	4.23	4.07
2006	3.56	2.38	3.48	3.99
2007	6.15	8.27	7.58	7.15
2008	4.57	3.97	2.59	0.69
2009	0.63	-0.94	-1.37	-0.88
2010	-2.04	-1.48	-0.56	-0.71
2011	-1.22	-1.06	-2.10	-2.78
2012	-2.87	-3.12	-3.41	-3.20
2013	-3.05	-3.73	-3.25	-2.44
2014	-1.92	0.34	1.25	1.98
2015	2.71	1.56	1.64	1.55
2016	1.80	2.02	1.93	1.52
2017	1.75	1.80	2.66	3.16
2018	2.45	3.62	3.24	3.12
2019	3.90	3.26	3.25	3.44
2020	2.42	-1.89	-0.80	-1.67
2021	-1.29	2.49	2.53	3.59
2022	5.05	9.21	11.58	12.89
2023	11.92	10.04	7.90	5.65

📊 Variación interanual (%) de Contratos_Temporales por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	5.73	4.29	7.01	6.94
2005	5.03	10.24	10.52	7.96
2006	10.07	6.92	3.91	3.63
2007	-0.50	-3.64	-4.81	-6.05
2008	-3.98	-7.48	-8.76	-13.27
2009	-21.00	-20.07	-17.97	-14.18
2010	-6.91	-3.17	-2.27	-2.21
2011	0.56	2.19	0.50	-2.05
2012	-8.24	-12.58	-13.17	-13.23
2013	-11.37	-6.60	-2.21	2.28
2014	4.97	6.48	4.56	5.30
2015	5.42	8.00	10.07	9.54
2016	10.12	5.46	6.21	5.89
2017	5.62	7.67	4.88	4.40
2018	4.38	3.59	3.51	3.93
2019	2.68	0.98	-0.70	-0.49
2020	-2.15	-21.12	-13.02	-9.03
2021	-6.80	20.18	13.93	8.92
2022	7.25	-6.20	-19.16	-27.03
2023	-26.17	-19.50	-11.53	-5.32

📊 Variación interanual (%) de Duracion_Media_Hipoteca por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	4.35	4.35	4.35	4.35
2005	4.17	4.17	4.17	4.17
2006	8.00	8.00	8.00	8.00
2007	3.70	3.70	3.70	3.70
2008	-3.57	-3.57	-3.57	-3.57
2009	-8.63	-11.11	-12.33	-11.11
2010	-2.72	1.37	4.22	0.00
2011	4.17	-2.71	-5.43	-2.79
2012	-6.68	1.39	1.46	0.00
2013	1.46	-8.33	-5.66	-2.83
2014	-2.83	4.55	3.00	1.46
2015	0.00	0.00	0.00	0.00
2016	1.43	0.00	2.91	1.43
2017	1.46	0.00	0.00	1.46
2018	1.39	2.91	1.39	1.39
2019	-4.17	1.39	0.00	-1.37
2020	0.00	0.00	-1.37	5.62
2021	7.26	2.79	2.79	-2.68
2022	-1.38	-1.38	0.00	1.40
2023	0.00	1.40	0.00	-2.72

📊 Variación interanual (%) de Euribor_12_Meses por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	-12.73	2.25	6.46	-2.10
2005	8.13	-4.41	-6.11	13.48
2006	27.24	51.35	64.43	46.99
2007	38.54	32.24	28.38	21.26
2008	9.56	15.57	15.38	-7.10
2009	-50.39	-66.89	-75.11	-71.52
2010	-44.91	-25.27	5.17	22.74
2011	41.29	69.83	50.62	35.01
2012	-3.35	-39.58	-57.81	-70.98
2013	-65.82	-60.58	-39.88	-11.02
2014	-1.52	12.51	-18.07	-36.98
2015	-54.68	-70.28	-63.46	-73.45
2016	-97.12	-110.04	-133.40	-183.83
2017	-1509.09	674.51	196.89	150.67
2018	84.19	41.52	7.74	-23.08
2019	-41.68	-22.00	89.90	94.65
2020	142.34	-22.94	7.67	72.52
2021	85.01	331.25	40.65	1.52
2022	-29.74	-179.50	-402.09	-678.10
2023	-1102.67	909.20	176.51	39.95

📊 Variación interanual (%) de Importe_Hipotecas por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	24.18	29.72	31.38	22.94
2005	20.97	31.41	30.04	31.40
2006	35.66	21.56	12.73	11.63
2007	5.77	0.49	-2.66	-12.98
2008	-31.46	-33.21	-40.38	-43.72
2009	-47.03	-38.54	-25.13	-18.65
2010	-0.01	-5.36	-4.04	-21.20
2011	-10.98	-41.68	-48.75	-42.84
2012	-49.92	-33.13	-31.34	-28.00
2013	-19.65	-34.00	-40.24	-25.50
2014	-24.00	2.89	35.85	22.28
2015	29.01	26.27	27.74	21.27
2016	15.27	29.41	6.34	21.71
2017	19.00	12.25	30.75	11.31
2018	12.88	20.97	17.32	21.26
2019	18.74	6.15	-13.98	10.46
2020	18.73	-16.55	-1.54	-0.49
2021	-11.49	47.05	58.50	29.16
2022	28.82	20.55	10.05	10.40
2023	-6.04	-24.34	-25.97	-21.92

📊 Variación interanual (%) de Interes_Fijo_Hipotecas por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	-16.70	-16.70	-16.70	-16.70
2005	-9.79	-9.79	-9.79	-9.79
2006	11.89	11.89	11.89	11.89
2007	23.09	23.09	23.09	23.09
2008	14.63	14.63	14.63	14.63
2009	-6.38	-6.38	-6.38	-6.38
2010	-10.66	-10.66	-10.66	-10.66
2011	-1.96	-1.96	-1.96	-1.96
2012	5.19	5.19	5.19	5.19
2013	7.40	7.40	7.40	7.40
2014	-1.41	-1.41	-1.41	-1.41
2015	-19.89	-19.89	-19.89	-19.89
2016	-23.49	-23.49	-23.49	-23.49
2017	-7.31	-7.31	-7.31	-7.31
2018	-4.42	-4.42	-4.42	-4.42
2019	0.66	0.66	0.66	0.66
2020	-24.26	-24.26	-24.26	-24.26
2021	-17.32	-17.32	-17.32	-17.32
2022	6.81	6.81	6.81	6.81
2023	61.76	61.76	61.76	61.76

📊 Variación interanual (%) de Interes_Variable_Hipotecas por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	-16.32	-16.32	-16.32	-16.32
2005	-6.96	-6.96	-6.96	-6.96
2006	12.28	12.28	12.28	12.28
2007	23.47	23.47	23.47	23.47
2008	10.15	10.15	10.15	10.15
2009	-15.49	-15.49	-15.49	-15.49
2010	-17.40	-17.40	-17.40	-17.40
2011	4.49	4.49	4.49	4.49
2012	9.68	9.68	9.68	9.68
2013	1.47	1.47	1.47	1.47
2014	-9.42	-9.42	-9.42	-9.42
2015	-13.87	-13.87	-13.87	-13.87
2016	-2.48	-2.48	-2.48	-2.48
2017	-12.38	-12.38	-12.38	-12.38
2018	-12.32	-12.32	-12.32	-12.32
2019	-5.37	-5.37	-5.37	-5.37
2020	-7.42	-7.42	-7.42	-7.42
2021	-8.49	-8.49	-8.49	-8.49
2022	-2.06	-2.06	-2.06	-2.06
2023	50.53	50.53	50.53	50.53

📊 Variación interanual (%) de Ipc_%_Variación_Anual por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	-41.23	10.34	8.89	25.30
2005	46.27	2.08	6.12	1.92
2006	24.49	22.45	2.88	-24.53
2007	-40.16	-39.17	-32.71	50.00
2008	84.93	91.78	105.56	-38.33
2009	-89.63	-115.00	-119.59	-90.54
2010	171.43	-433.33	-313.79	1000.00
2011	155.26	41.43	40.32	6.49
2012	-41.24	-42.42	-3.45	15.85
2013	45.61	-3.51	-52.38	-93.68
2014	-97.59	-90.91	-130.00	-400.00
2015	-1850.00	-300.00	33.33	-22.22
2016	-31.43	220.00	-37.50	-271.43
2017	-433.33	-293.75	-650.00	95.83
2018	-60.00	-11.29	23.64	10.64
2019	6.25	-43.64	-82.35	-71.15
2020	-38.24	-161.29	-258.33	-253.33
2021	-28.57	-463.16	-636.84	-860.87
2022	1473.33	288.41	196.08	11.43
2023	-36.44	-69.03	-74.17	-48.21

📊 Variación interanual (%) de Ipi_Variacion_Anual_Corregida por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	30.23	89.80	253.85	-94.23
2005	-80.36	-102.15	-92.39	1633.33
2006	1000.00	-5100.00	1185.71	132.69
2007	-21.49	-8.00	-41.11	-87.60
2008	-133.68	-264.13	-528.30	-3226.67
2009	1981.25	287.42	89.87	-62.26
2010	-104.35	-115.04	-97.22	-108.47
2011	34.48	-151.14	225.00	-1126.67
2012	-533.33	344.44	400.00	28.57
2013	-10.06	-56.00	-82.05	-129.29
2014	-123.03	-186.36	-174.29	-39.66
2015	77.14	17.11	450.00	180.00
2016	104.84	-46.07	-100.00	-51.02
2017	-62.99	47.92	inf	208.33
2018	61.70	-47.89	-74.07	-143.92
2019	-98.68	-18.92	-42.86	-146.15
2020	-16000.00	-2526.67	-1283.33	-343.33
2021	-161.64	-215.80	-145.77	-168.49
2022	-70.41	-81.49	78.46	-112.00
2023	-58.62	-155.13	-170.69	466.67

📊 Variación interanual (%) de Ipi_Variacion_Annual_Original por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	-46.55	-670.59	292.31	-114.29
2005	-208.06	-22.68	-90.20	-412.50
2006	-453.73	-72.00	630.00	340.00
2007	-55.27	304.76	-23.29	-57.27
2008	-216.04	-185.88	-423.21	-1089.36
2009	406.50	790.41	122.10	-60.65
2010	-99.84	-117.38	-100.00	-101.64
2011	-6800.00	-166.37	-inf	-6433.33
2012	-344.78	166.67	973.68	-2.11
2013	38.41	-92.50	-100.49	-127.42
2014	-146.70	-186.67	2200.00	-27.45
2015	-64.15	761.54	547.83	151.35
2016	134.21	-4.46	-108.05	-63.44
2017	66.29	-103.74	-566.67	329.41
2018	-83.78	-2625.00	-69.64	-126.03
2019	-12.50	-122.77	223.53	-165.79
2020	-900.00	3039.13	-332.73	-348.00
2021	-136.90	-221.47	-135.16	-190.32
2022	11.29	-84.72	131.11	-153.57
2023	-34.78	-164.93	-212.50	33.33

📊 Variación interanual (%) de Mro_Rate por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	0.00	0.00	0.00	0.00
2005	0.00	0.00	0.00	0.00
2006	0.00	0.00	0.00	0.00
2007	0.00	0.00	0.00	0.00
2008	0.00	0.00	0.00	-10.48
2009	-51.98	-66.01	-67.62	-64.27
2010	-40.50	-15.94	-11.75	-10.68
2011	10.00	25.00	39.95	24.46
2012	-12.84	-23.30	-42.33	-35.59
2013	-28.73	-39.26	-34.68	-47.12
2014	-67.80	-65.57	-73.40	-77.95
2015	-77.27	-75.07	-64.34	-46.51
2016	-47.25	-90.11	-90.22	-90.22
2017	-100.00	-100.00	-100.00	-100.00
2018	NaN	NaN	NaN	NaN
2019	NaN	NaN	NaN	NaN
2020	NaN	NaN	NaN	NaN
2021	NaN	NaN	NaN	NaN
2022	inf	inf	inf	inf
2023	1519.44	1705.33	607.14	218.44

📊 Variación interanual (%) de Numero_Compraventas por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	3.55	20.68	7.45	23.40
2005	3.15	7.82	7.58	6.20
2006	18.95	4.25	2.44	0.34
2007	-1.25	-9.57	-15.84	-22.64
2008	-31.06	-31.00	-34.08	-34.70
2009	-34.19	-23.61	-12.54	4.88
2010	2.27	27.70	-25.09	14.41
2011	-30.47	-40.75	-4.99	-28.65
2012	-6.76	-7.13	-1.60	25.36
2013	-21.01	-3.34	-6.25	-30.42
2014	48.66	12.64	13.86	19.53
2015	5.02	16.64	16.98	2.76
2016	21.01	15.32	10.75	10.02
2017	20.43	16.46	14.42	14.24
2018	8.56	12.25	10.62	6.70
2019	2.17	-7.27	-6.17	2.63
2020	-16.15	-47.26	6.81	1.24
2021	22.25	125.55	22.15	20.47
2022	23.64	12.12	4.59	-9.90
2023	-9.91	-14.54	-15.71	-3.59

📊 Variación interanual (%) de Numero_Hipotecas por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	7.42	14.31	16.59	9.92
2005	7.11	16.49	15.12	15.60
2006	20.70	5.27	0.67	0.75
2007	-4.06	-5.50	-7.03	-15.15
2008	-29.95	-29.19	-34.47	-37.73
2009	-36.86	-26.85	-10.52	-7.39
2010	5.07	-3.85	-8.92	-19.23
2011	-12.77	-37.89	-43.23	-39.03
2012	-43.90	-29.26	-26.43	-24.94
2013	-16.61	-29.21	-38.11	-26.24
2014	-24.27	-0.79	29.21	22.19
2015	24.02	20.57	23.54	14.87
2016	14.08	24.93	0.95	19.64
2017	11.56	5.42	24.00	2.72
2018	6.74	14.03	11.37	13.45
2019	16.48	4.45	-15.81	11.58
2020	5.26	-19.41	-5.65	-6.58
2021	-7.30	37.69	52.51	24.79
2022	20.03	13.58	5.42	6.04
2023	-5.99	-21.86	-24.32	-19.36

📊 Variación interanual (%) de Pib_Trimestral_Ajustado por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	7.00	6.71	7.15	7.72
2005	7.49	8.00	8.09	8.07
2006	8.46	8.54	8.36	7.79
2007	8.29	7.45	6.37	6.83
2008	5.38	4.42	3.04	0.23
2009	-3.05	-4.13	-3.75	-3.25
2010	-0.28	0.14	0.51	1.18
2011	0.02	0.20	-0.86	-2.48
2012	-2.39	-3.33	-2.97	-3.56
2013	-2.11	-1.36	-1.04	0.57
2014	0.20	0.82	1.53	2.65
2015	4.03	4.66	5.03	4.82
2016	3.48	3.14	3.36	3.21
2017	3.73	4.45	3.85	4.73
2018	3.68	3.50	3.80	3.46
2019	4.30	3.38	2.84	3.17
2020	-3.09	-20.71	-8.09	-7.86
2021	-0.95	21.73	7.97	11.08
2022	11.09	12.44	11.01	10.27
2023	11.26	9.35	8.50	7.36

📊 Variación interanual (%) de Pib_Trimestral_No_Ajustado por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	6.78	7.10	7.12	7.55
2005	7.82	8.26	7.73	7.83
2006	8.51	8.15	8.13	8.33
2007	7.60	7.39	6.56	7.33
2008	4.61	4.62	3.49	0.47
2009	-3.59	-4.47	-3.34	-2.79
2010	-0.14	0.57	0.49	0.60
2011	0.53	-0.31	-0.81	-2.43
2012	-2.47	-3.66	-2.37	-3.69
2013	-2.13	-1.21	-0.77	0.09
2014	0.62	0.70	1.56	2.27
2015	3.92	4.62	4.80	5.15
2016	3.41	3.49	3.40	2.91
2017	3.85	4.23	4.13	4.53
2018	3.61	3.69	3.38	3.75
2019	4.37	3.47	2.96	2.94
2020	-3.45	-20.66	-7.84	-7.44
2021	-2.00	21.59	8.21	11.35
2022	12.33	12.10	11.24	9.32
2023	11.44	9.38	8.16	7.60

📊 Variación interanual (%) de Poblacion por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	1.72	1.65	1.57	1.67
2005	1.76	1.82	1.87	1.76
2006	1.65	1.62	1.60	1.68
2007	1.76	1.87	1.97	1.97
2008	1.97	1.81	1.65	1.45
2009	1.25	1.04	0.84	0.68
2010	0.53	0.48	0.42	0.40
2011	0.39	0.38	0.37	0.35
2012	0.32	0.18	0.03	-0.10
2013	-0.23	-0.29	-0.36	-0.41
2014	-0.46	-0.39	-0.32	-0.23
2015	-0.15	-0.13	-0.10	-0.06
2016	-0.01	0.04	0.09	0.13
2017	0.17	0.17	0.18	0.25
2018	0.32	0.38	0.44	0.51
2019	0.59	0.69	0.80	0.82
2020	0.85	0.70	0.55	0.36
2021	0.17	0.05	0.00	0.11
2022	0.18	0.53	0.92	1.09
2023	1.26	1.25	1.13	1.14

📊 Variación interanual (%) de Precio_M2_Vivienda por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	18.36	17.50	16.80	17.22
2005	15.74	13.91	13.41	12.75
2006	12.00	10.81	9.83	9.11
2007	7.24	5.78	5.34	4.77
2008	3.81	2.01	0.36	-3.21
2009	-6.82	-8.34	-8.31	-6.25
2010	-4.72	-3.75	-3.42	-3.53
2011	-4.72	-5.24	-5.61	-6.78
2012	-7.22	-8.32	-9.47	-10.02
2013	-8.06	-6.45	-4.49	-4.20
2014	-3.76	-2.89	-2.64	-0.26
2015	-0.10	1.20	1.39	1.85
2016	2.37	2.00	1.61	1.47
2017	2.24	1.57	2.69	3.09
2018	2.67	3.78	3.21	3.86
2019	4.45	3.12	3.07	2.10
2020	0.25	-1.67	-1.14	-1.85
2021	-0.91	2.43	2.59	4.43
2022	6.68	5.55	4.72	3.25
2023	3.14	3.02	4.16	5.32

📊 Variación interanual (%) de Renta_Disponible_Bruta por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	5.98	5.78	5.04	6.34
2005	5.39	8.44	7.85	5.91
2006	6.56	5.96	5.65	6.12
2007	5.14	5.22	0.85	9.79
2008	4.32	5.67	7.10	4.39
2009	-0.76	0.01	0.47	-1.94
2010	1.90	0.40	-0.23	0.44
2011	0.62	1.58	1.67	-1.16
2012	-3.38	-6.59	-4.74	-5.72
2013	-1.62	0.86	-1.65	0.90
2014	-2.78	-1.63	0.71	4.10
2015	5.55	5.63	4.12	1.50
2016	2.79	3.74	0.89	2.86
2017	1.90	3.49	2.44	4.08
2018	3.08	2.03	3.28	3.27
2019	4.99	7.05	3.78	3.97
2020	2.53	-8.48	1.37	-2.72
2021	0.87	7.75	3.83	6.73
2022	5.33	5.29	3.94	6.31
2023	9.59	11.76	11.66	9.71

 Variación interanual (%) de Tasa_Paro_(%) por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	2003	NaN	NaN	NaN	NaN
2004	-4.09	-1.68	-4.96	-7.39	
2005	-11.57	-15.96	-21.69	-17.28	
2006	-11.21	-9.44	-3.92	-5.17	
2007	-6.76	-6.04	-0.87	3.75	
2008	14.01	30.64	40.20	60.91	
2009	79.58	71.53	58.06	35.32	
2010	15.08	11.93	10.37	7.77	
2011	6.25	3.77	8.63	12.18	
2012	14.75	18.22	16.49	14.23	
2013	11.37	6.80	3.47	-0.16	
2014	-3.75	-6.10	-7.72	-7.89	
2015	-8.29	-8.58	-10.52	-11.81	
2016	-11.69	-10.59	-10.72	-10.86	
2017	-10.71	-13.90	-13.38	-11.16	
2018	-10.72	-11.27	-11.17	-12.69	
2019	-12.19	-8.25	-4.33	-4.64	
2020	-1.97	9.34	16.81	17.05	
2021	12.01	0.39	-9.53	-16.68	
2022	-14.93	-17.54	-13.46	-3.35	
2023	-2.55	-8.04	-6.60	-9.16	

📊 Variación interanual (%) de Ti_Credito por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	Q1	Q2	Q3	Q4
2003	NaN	NaN	NaN	NaN
2004	-17.01	-15.60	-3.30	-1.80
2005	-1.17	3.44	-3.23	-0.99
2006	9.65	20.10	34.36	39.66
2007	33.53	27.82	24.20	19.66
2008	12.45	9.33	13.49	9.42
2009	-20.15	-40.11	-51.96	-55.87
2010	-40.90	-25.95	-11.34	0.84
2011	17.32	38.34	39.84	37.46
2012	23.81	1.26	-10.16	-18.48
2013	-16.02	-8.86	-5.86	1.90
2014	1.17	-0.81	-1.93	-9.53
2015	-21.16	-26.14	-26.21	-23.67
2016	-18.22	-9.42	-7.21	-5.19
2017	-2.38	-5.75	-1.31	-3.02
2018	-0.44	0.76	-1.51	5.15
2019	7.68	9.60	-0.56	-12.16
2020	-14.33	-17.59	-10.26	-8.00
2021	-14.70	-14.68	-15.70	-11.74
2022	-2.91	9.19	36.91	91.87
2023	131.18	130.96	92.93	40.48

📊 Variación interanual (%) de Tipo_Interes_Medio por trimestre:

Trimestre_simple Q1 Q2 Q3 Q4

Año	2003	NaN	NaN	NaN	NaN
2004	-19.89	-17.37	-16.46	-10.04	
2005	-9.40	-7.43	-6.34	-5.34	
2006	-0.29	6.95	17.90	25.20	
2007	29.15	26.44	22.64	17.71	
2008	14.44	8.54	9.51	9.58	
2009	2.86	-12.67	-24.01	-27.53	
2010	-28.34	-16.87	-12.27	-8.53	
2011	-5.50	2.94	13.70	14.78	
2012	16.97	10.62	2.36	4.84	
2013	1.47	3.63	4.61	-0.94	
2014	-1.29	-8.64	-10.86	-15.80	
2015	-18.71	-15.33	-12.71	-7.41	
2016	-1.80	-3.92	-2.22	-3.95	
2017	-3.36	-7.43	-13.86	-14.24	
2018	-16.14	-10.29	-6.24	-3.08	
2019	-1.51	-1.64	-1.79	-4.82	
2020	-12.26	-17.82	-16.69	-14.13	
2021	-12.81	-7.80	-13.15	-12.73	
2022	-9.68	-8.63	5.77	30.96	
2023	56.75	74.07	65.93	34.51	

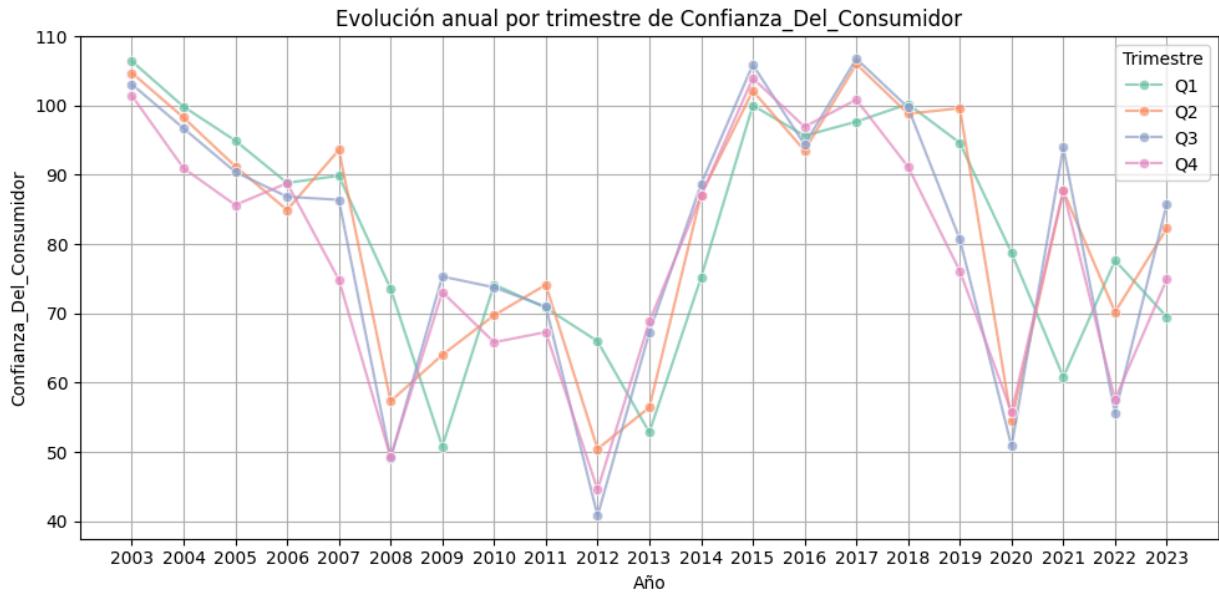
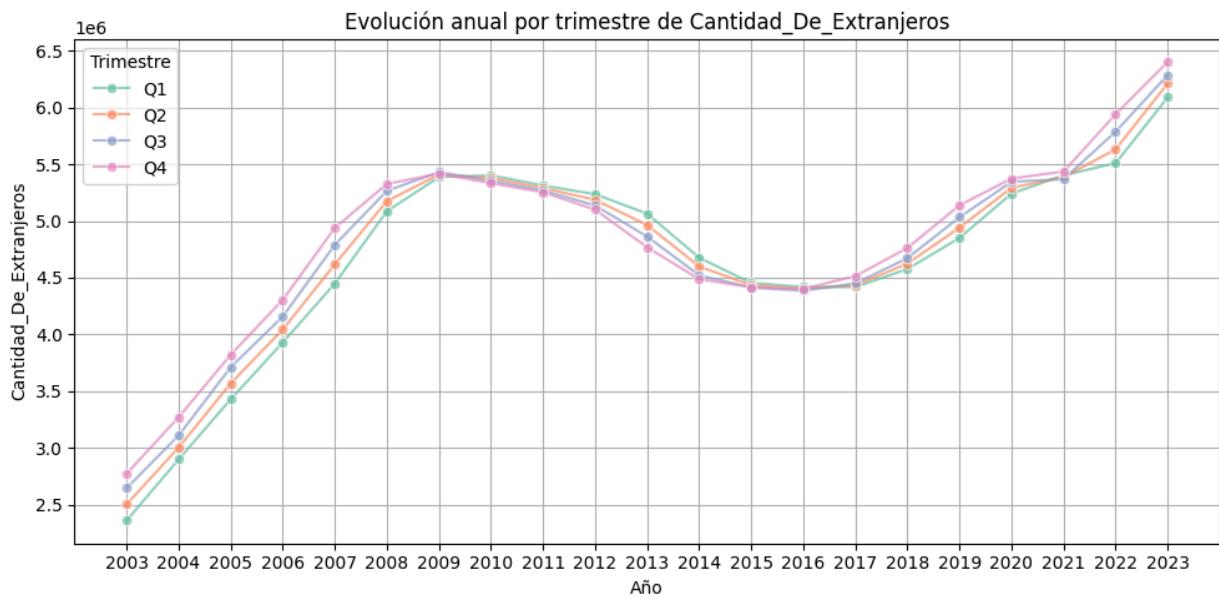
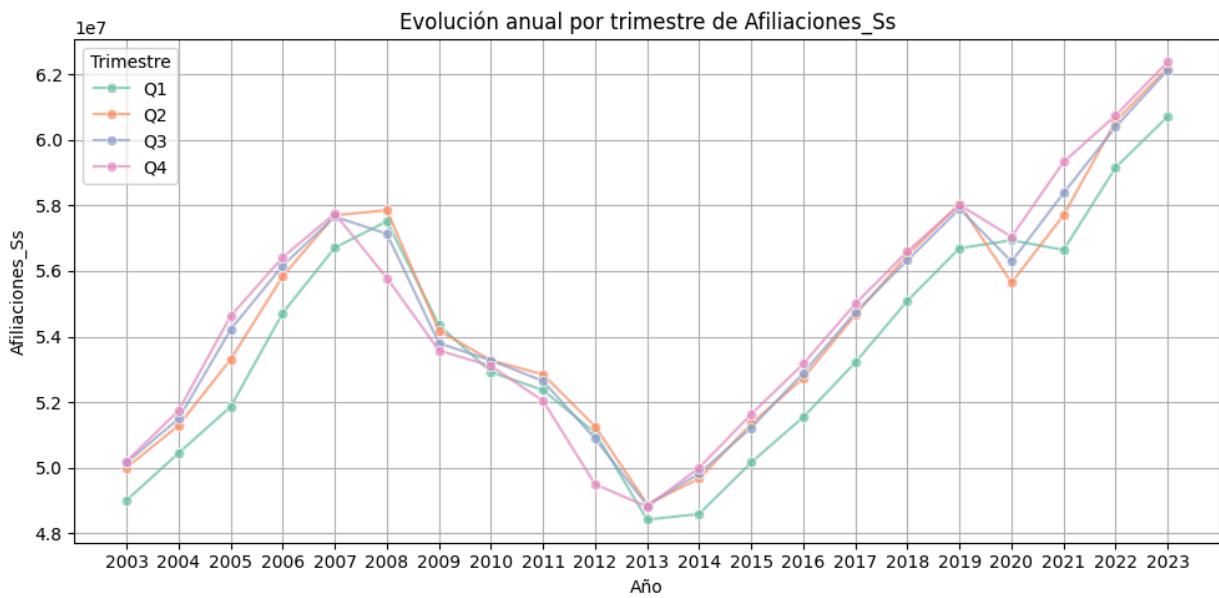
📊 Variación interanual (%) de Valor_Compraventa por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	16.09	43.06	25.38	42.73
2005	22.91	22.87	23.41	24.44
2006	37.24	18.95	15.63	13.14
2007	3.97	-4.24	-13.50	-22.49
2008	-30.22	-33.16	-35.21	-38.08
2009	-38.17	-27.31	-15.41	3.97
2010	4.41	34.24	-27.41	15.14
2011	-37.49	-47.95	-11.03	-38.37
2012	-13.67	-15.81	-13.49	13.20
2013	-29.02	-9.58	-7.53	-29.85
2014	55.76	16.38	12.35	19.94
2015	2.11	16.46	21.45	3.99
2016	27.05	18.28	12.52	12.85
2017	25.16	20.57	22.18	18.17
2018	11.16	16.50	13.92	12.73
2019	6.57	-3.60	0.23	8.79
2020	-10.25	-42.04	7.50	4.82
2021	28.86	132.19	34.86	29.36
2022	33.70	18.61	7.03	-10.63
2023	-13.16	-15.46	-15.45	-1.55

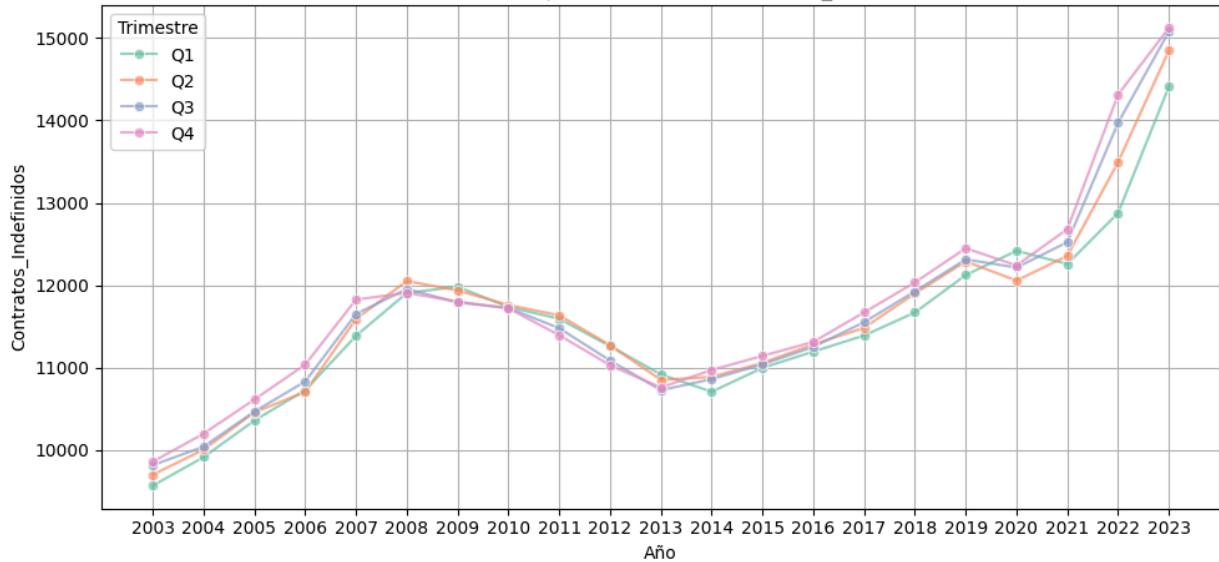
📊 Variación interanual (%) de Valor_Medio_Hipotecas por trimestre:

Trimestre_simple	Q1	Q2	Q3	Q4
Año				
2003	NaN	NaN	NaN	NaN
2004	15.60	13.48	12.68	11.85
2005	12.94	12.81	12.96	13.67
2006	12.39	15.48	11.98	10.80
2007	10.24	6.34	4.70	2.55
2008	-2.15	-5.67	-9.02	-9.61
2009	-16.10	-15.99	-16.33	-12.15
2010	-4.84	-1.57	5.35	-2.44
2011	2.05	-6.10	-9.72	-6.25
2012	-10.73	-5.47	-6.67	-4.07
2013	-3.65	-6.78	-3.44	1.00
2014	0.35	3.71	5.14	0.07
2015	4.02	4.73	3.39	5.58
2016	1.04	3.59	5.34	1.73
2017	6.67	6.48	5.44	8.36
2018	5.75	6.09	5.35	6.89
2019	1.94	1.62	2.17	-1.00
2020	12.79	3.54	4.36	6.51
2021	-4.51	6.80	3.93	3.50
2022	7.32	6.14	4.39	4.11
2023	-0.06	-3.17	-2.18	-3.17

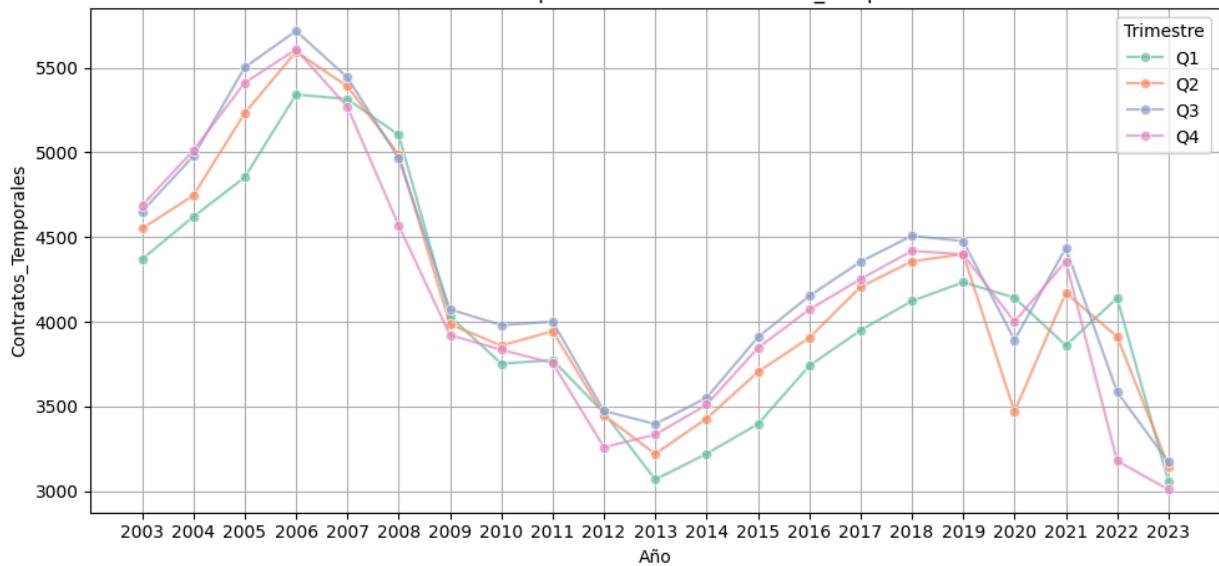
```
In [19]: for col in cols:
    plt.figure(figsize=(10, 5))
    sns.lineplot(
        data=df_clean,
        x='Año',
        y=col,
        hue='Trimestre_simple',
        marker='o',
        palette='Set2',
        alpha=0.7
    )
    plt.title(f'Evolución anual por trimestre de {col}')
    plt.xlabel('Año')
    plt.ylabel(col)
    plt.grid(True)
    plt.legend(title='Trimestre')
    plt.xticks(df_clean['Año'].unique())
    plt.tight_layout()
    plt.show()
```



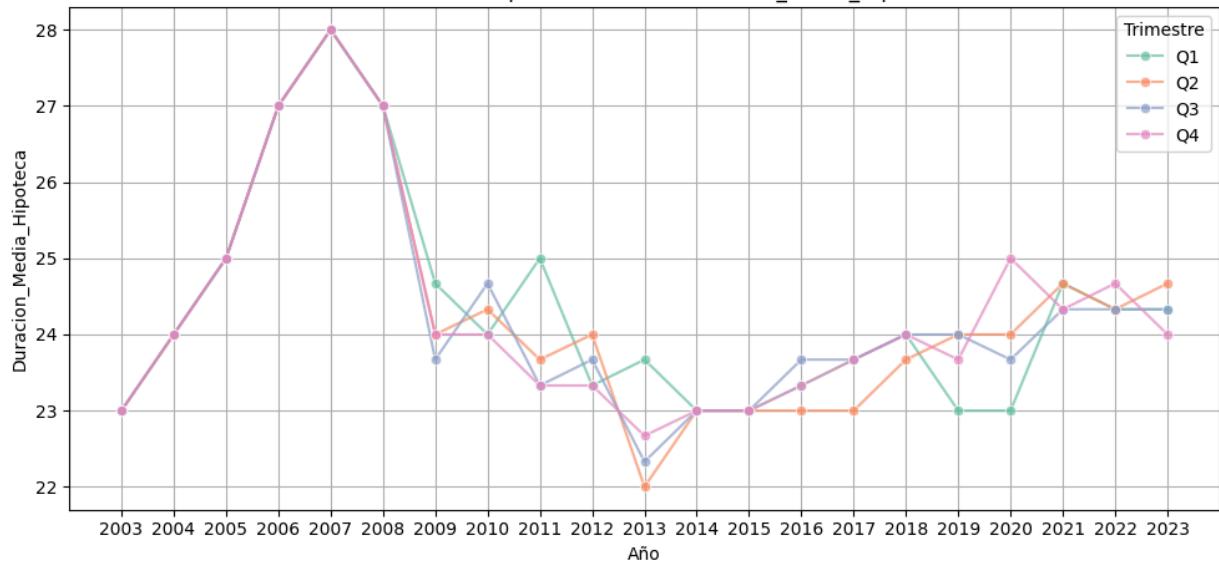
Evolución anual por trimestre de Contratos_Indefinidos



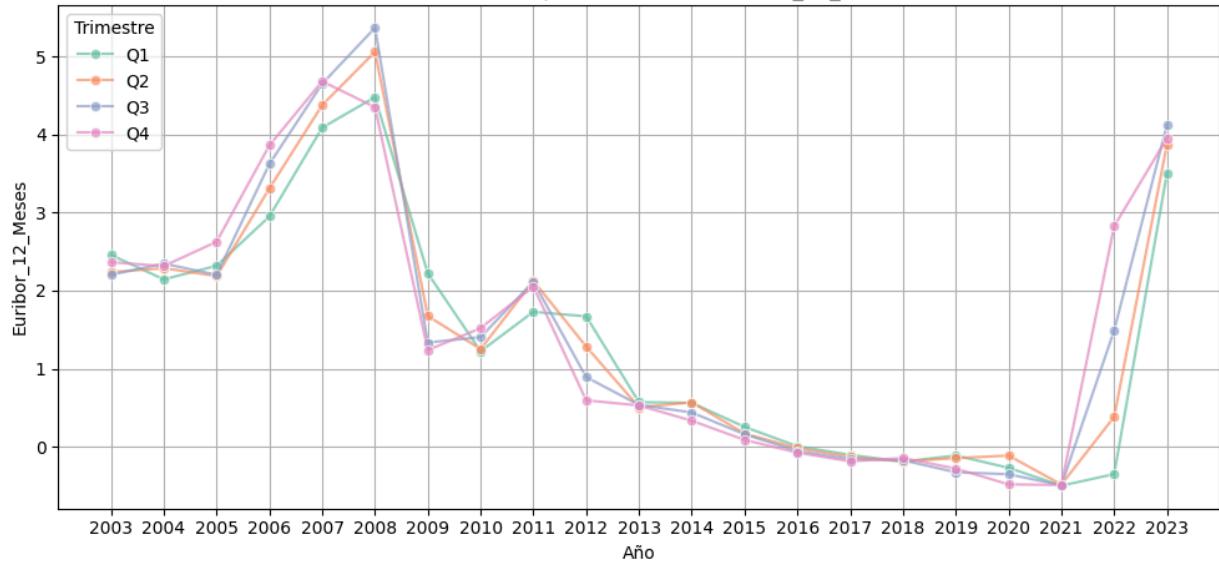
Evolución anual por trimestre de Contratos_Temporales



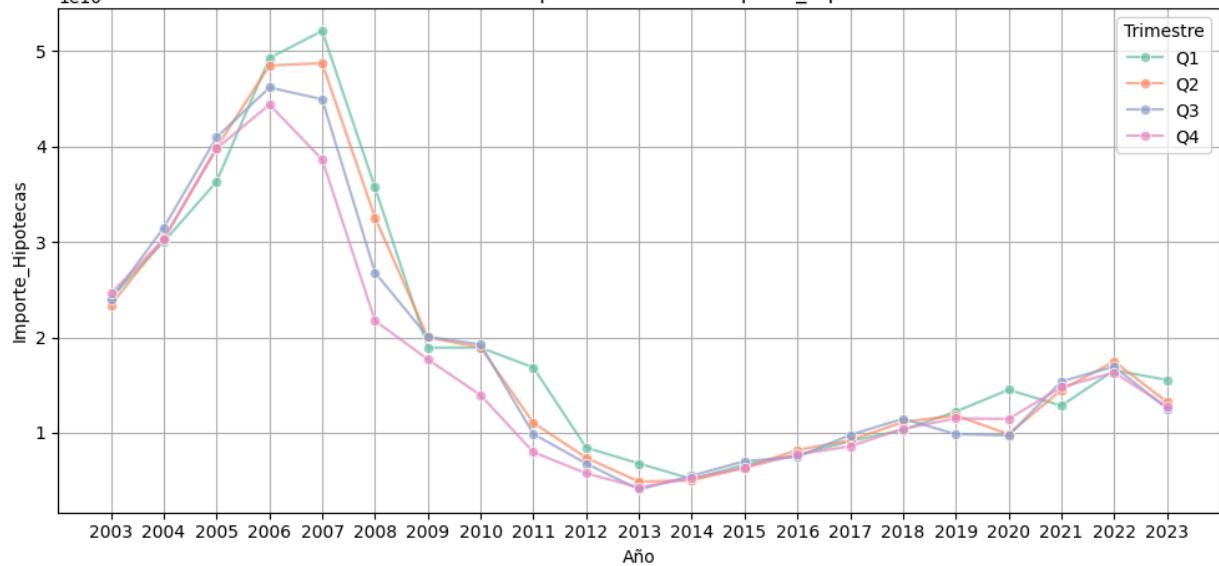
Evolución anual por trimestre de Duracion_Media_Hipoteca



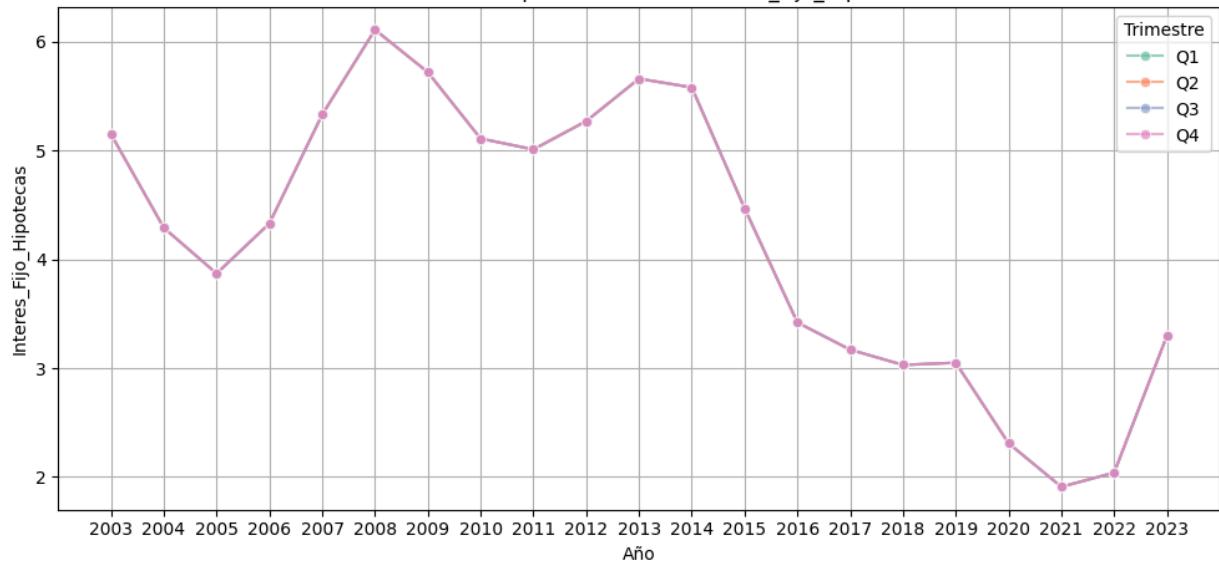
Evolución anual por trimestre de Euribor_12_Meses



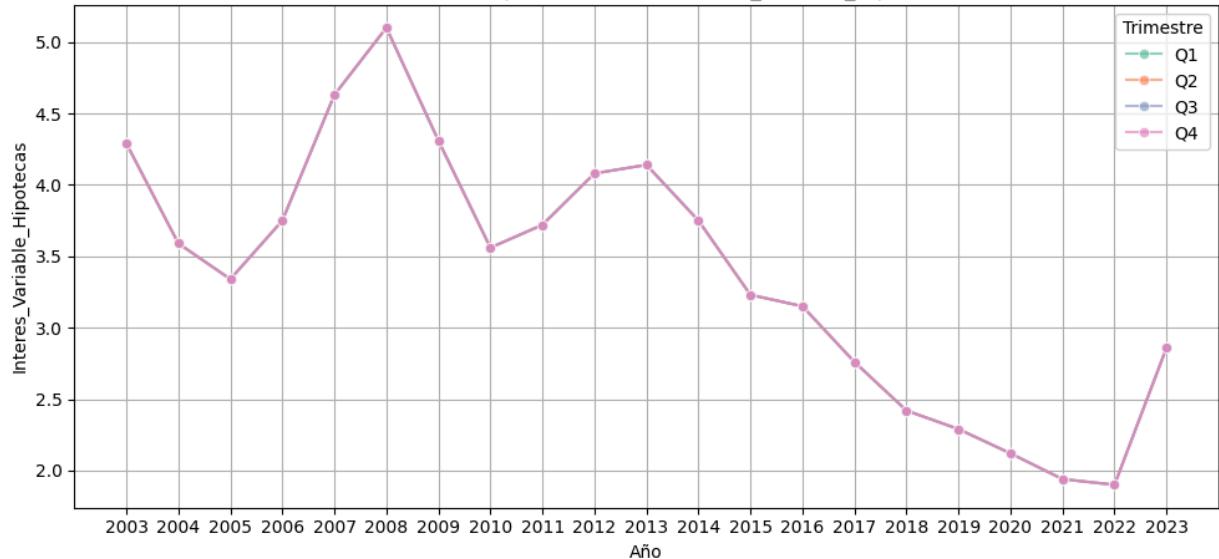
Evolución anual por trimestre de Importe_Hipotecas



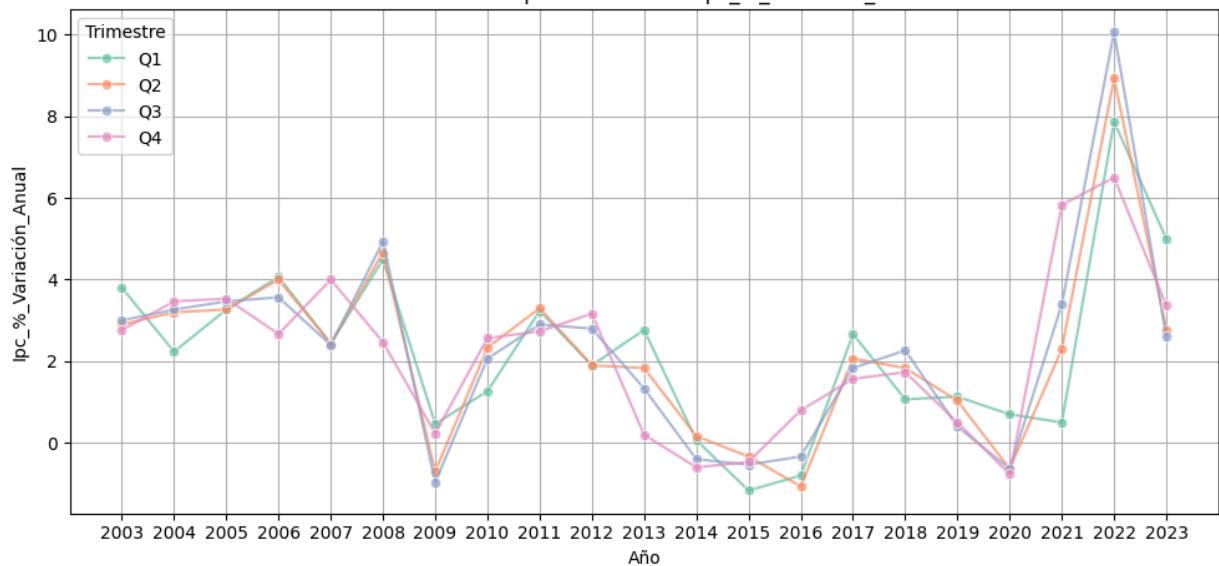
Evolución anual por trimestre de Interes_Fijo_Hipotecas



Evolución anual por trimestre de Interes_Variable_Hipotecas



Evolución anual por trimestre de lpc_%_Variación_Annual



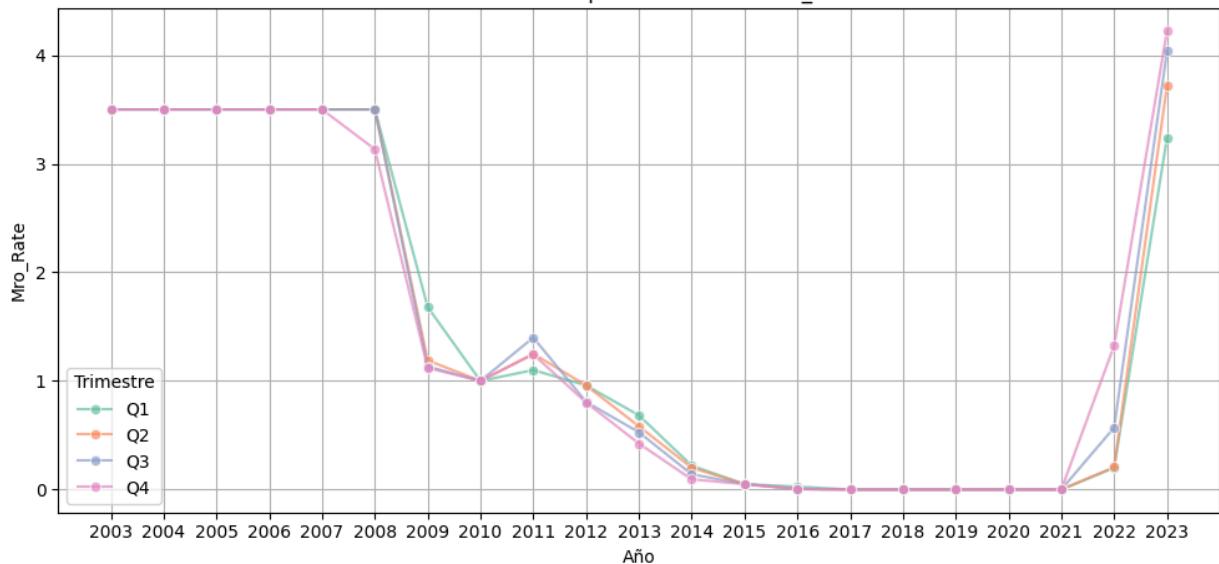
Evolución anual por trimestre de Ipi_Variacion_Annual_Corregida



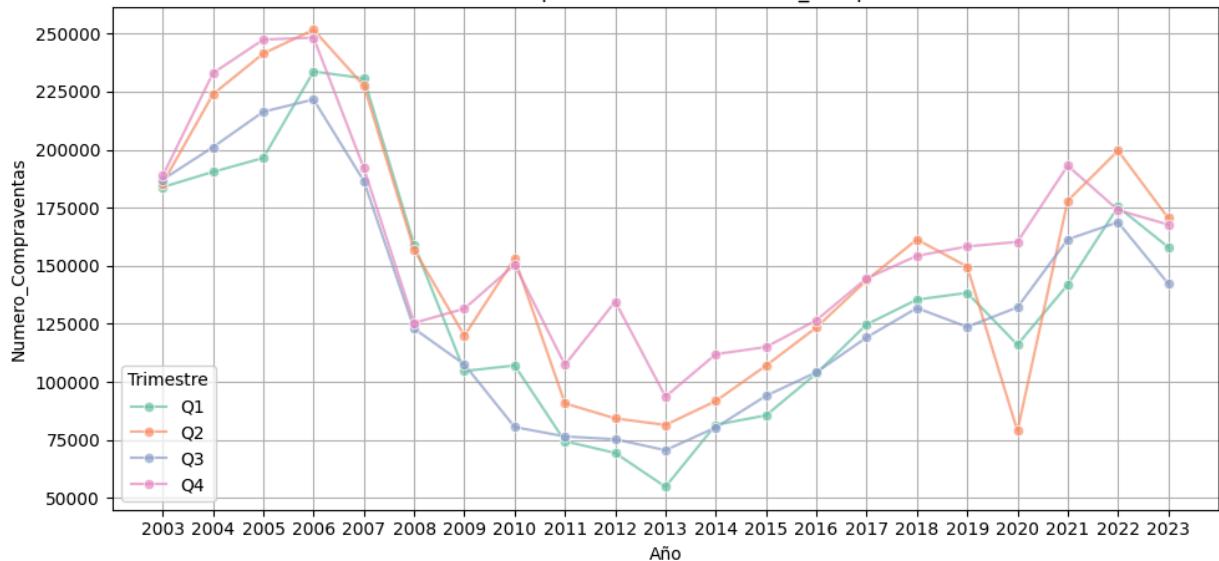
Evolución anual por trimestre de Ipi_Variacion_Annual_Original



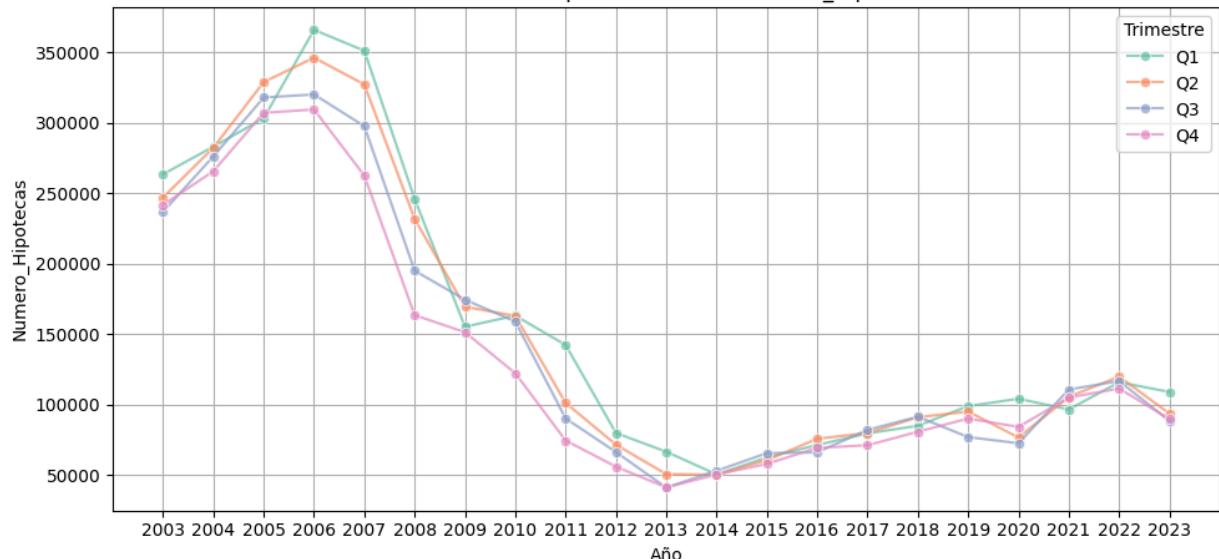
Evolución anual por trimestre de Mro_Rate



Evolución anual por trimestre de Numero_Compraventas



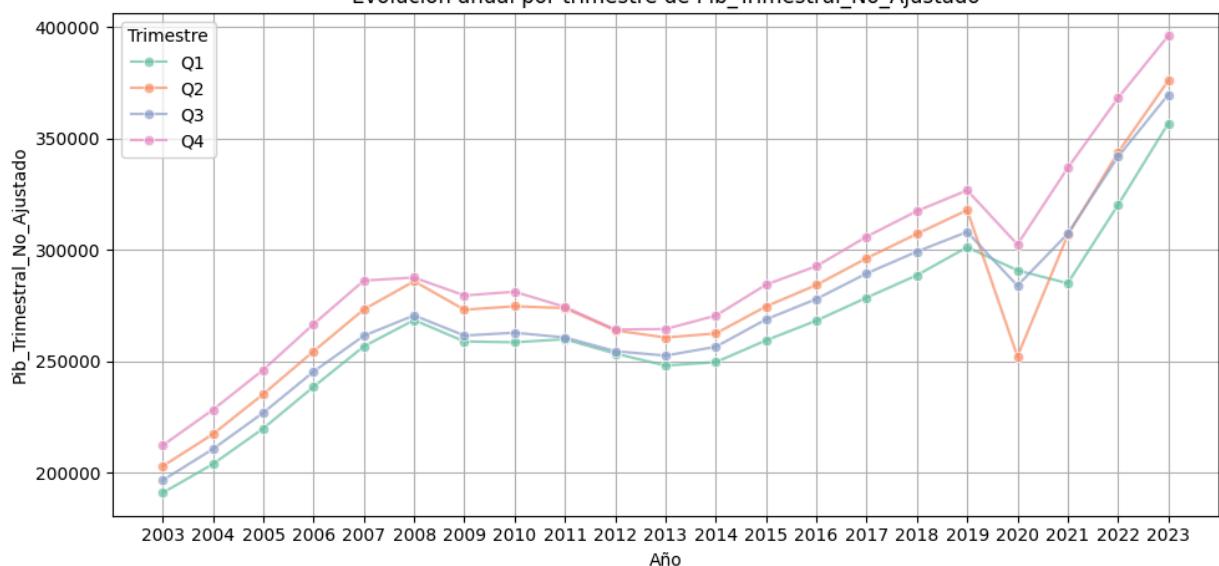
Evolución anual por trimestre de Numero_Hipotecas

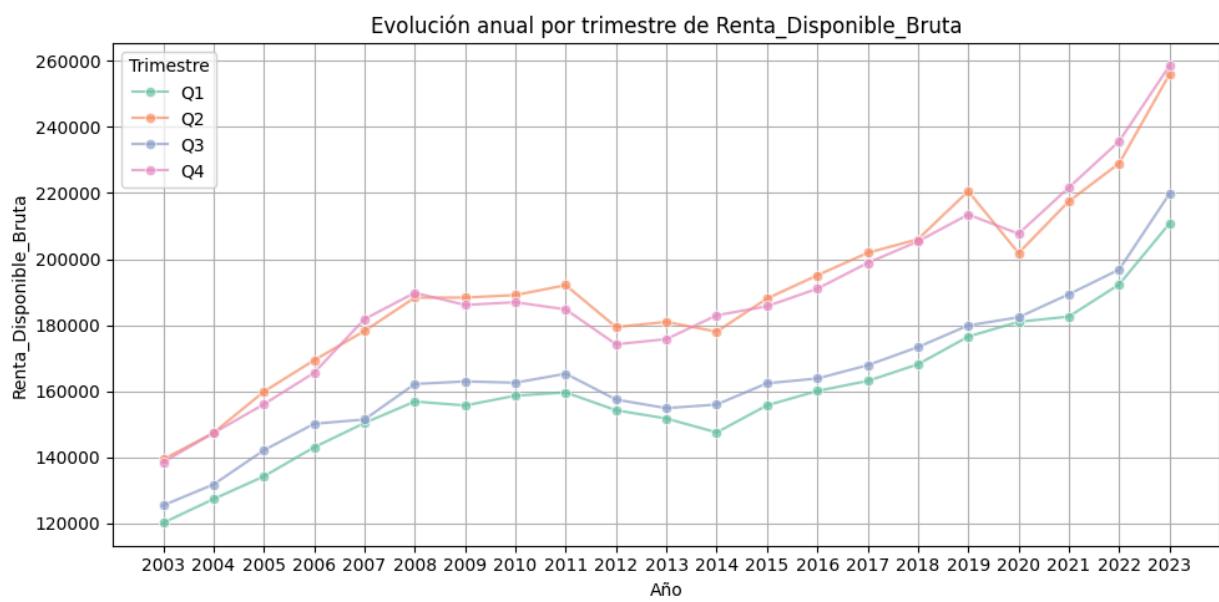
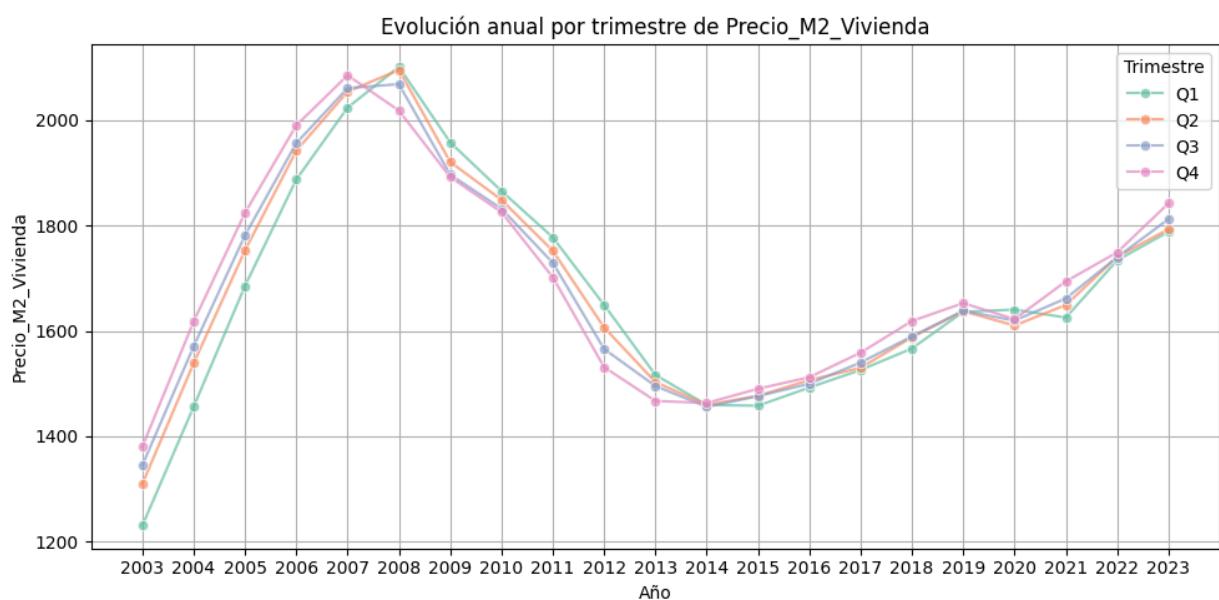
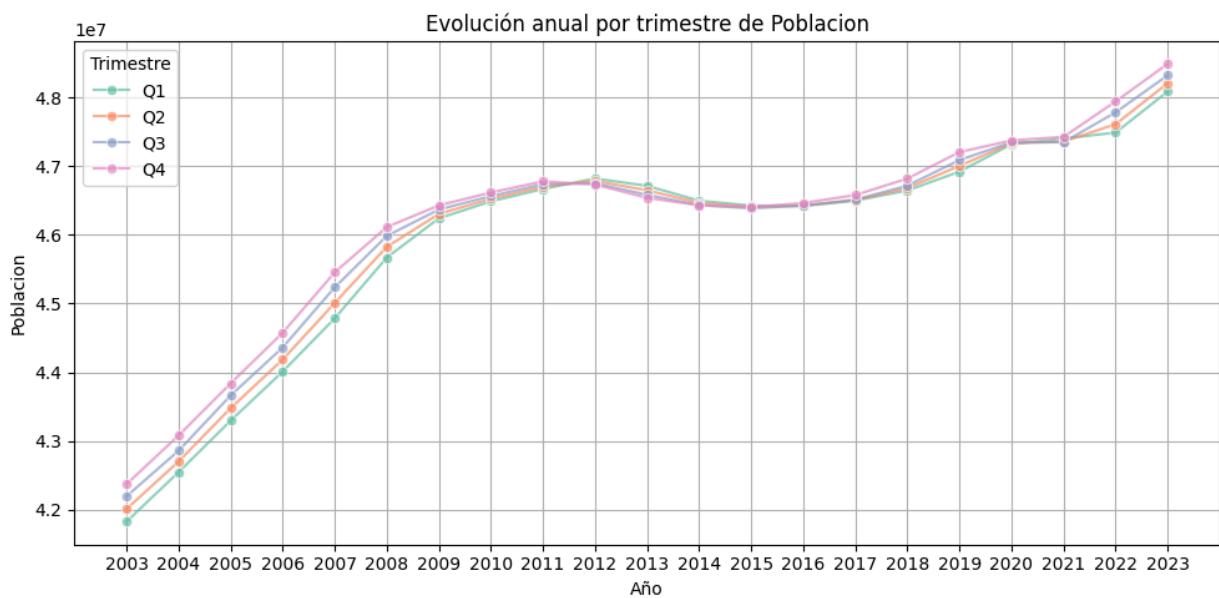


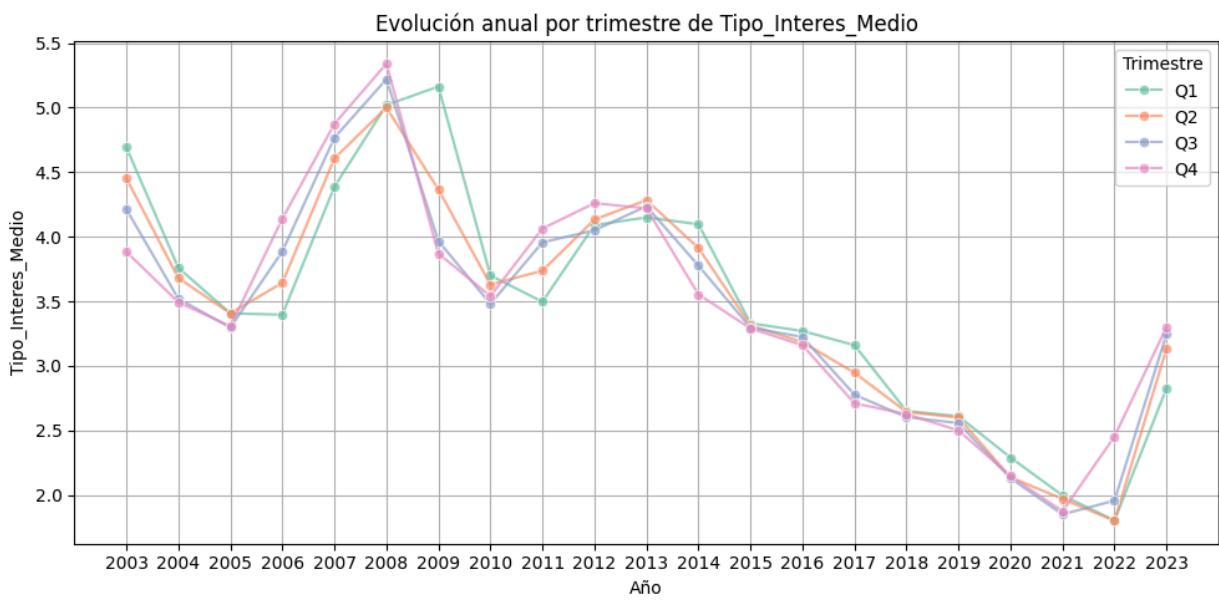
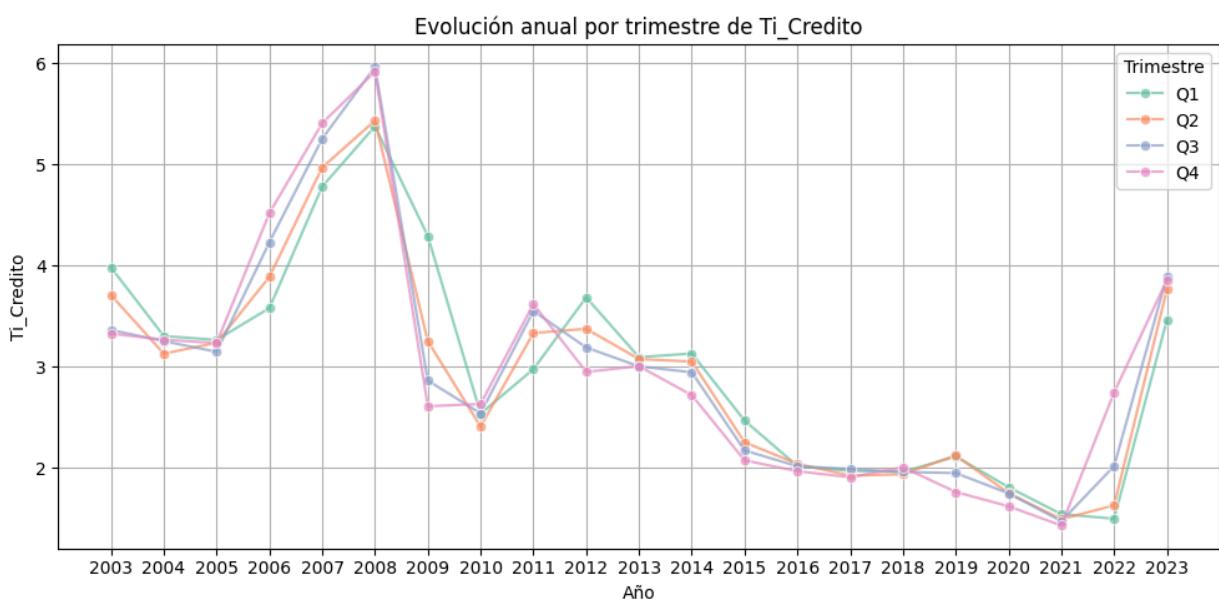
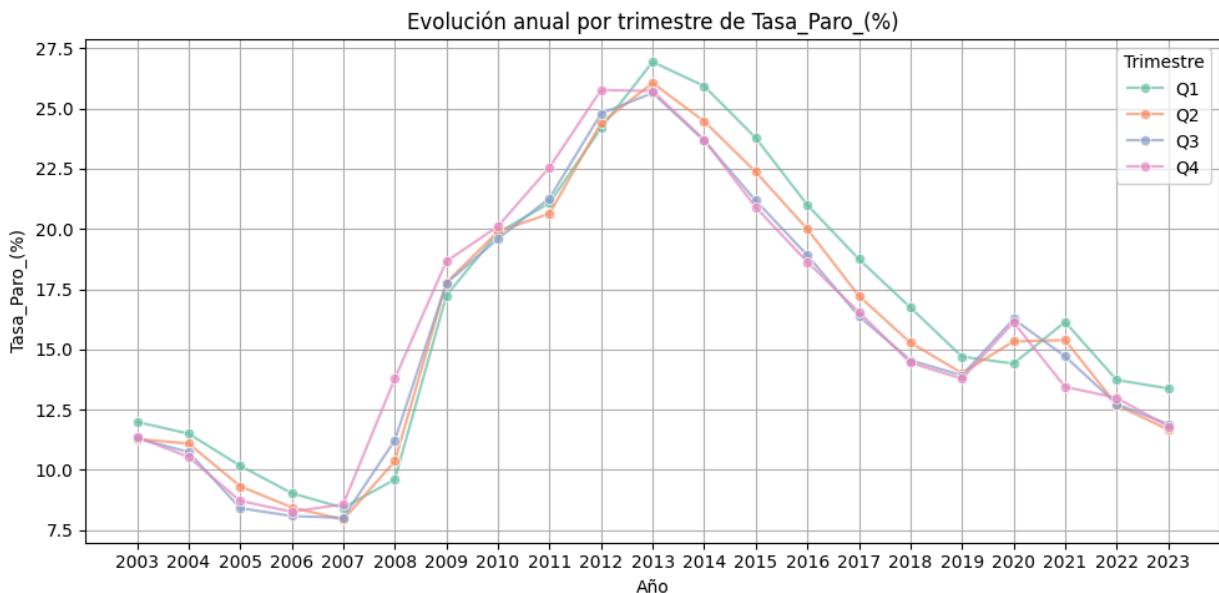
Evolución anual por trimestre de Pib_Trimestral_Ajustado

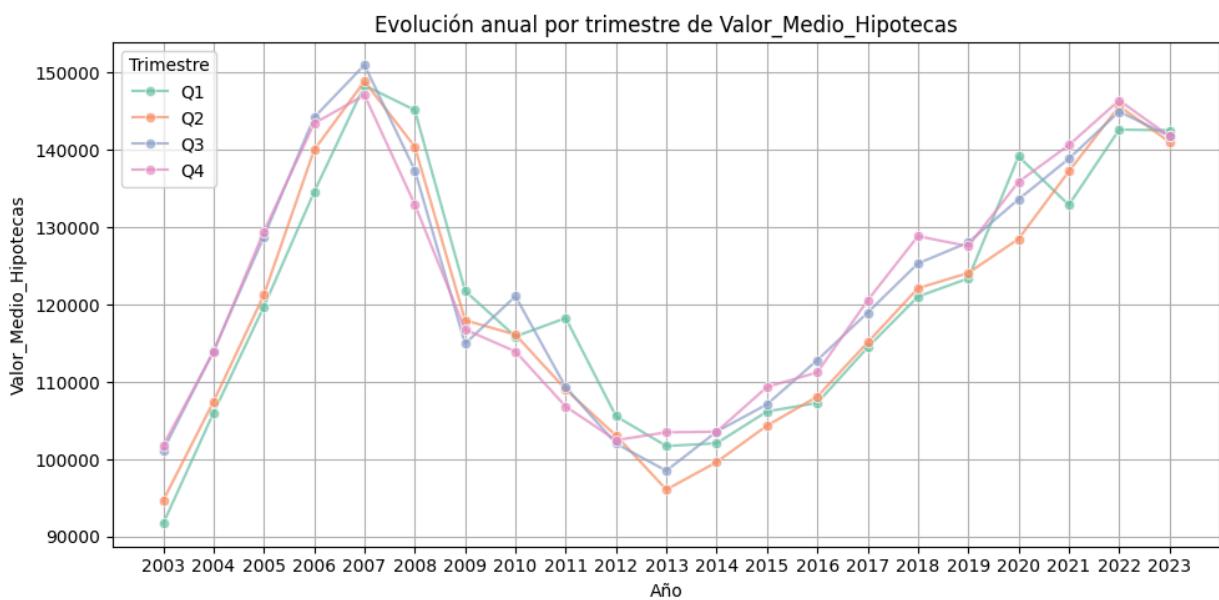
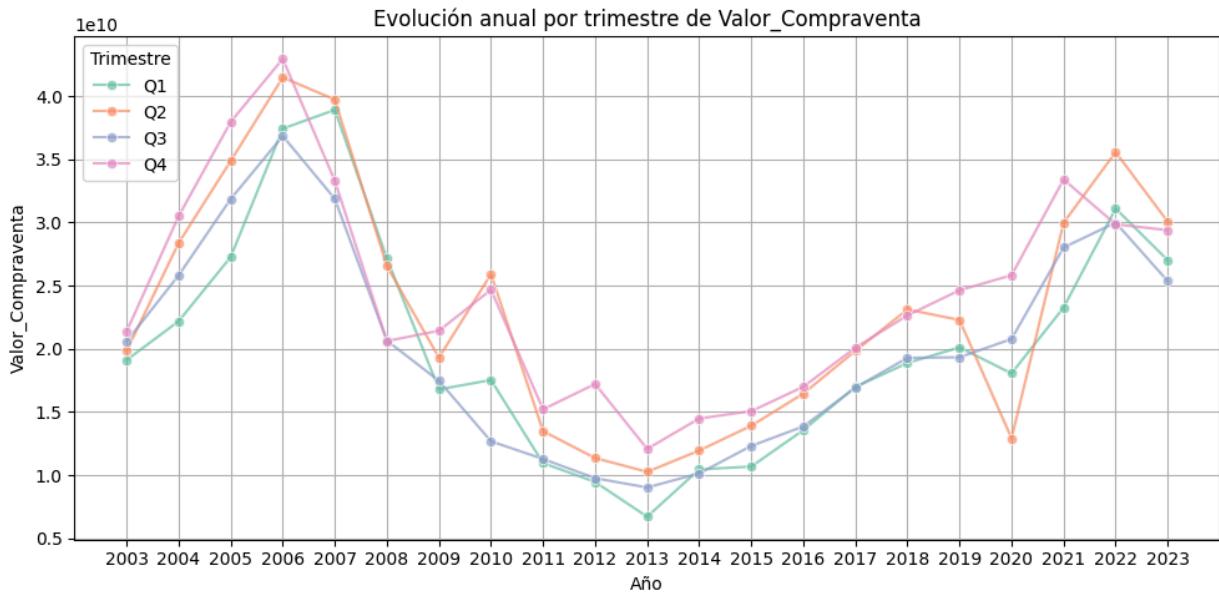


Evolución anual por trimestre de Pib_Trimestral_No_Ajustado









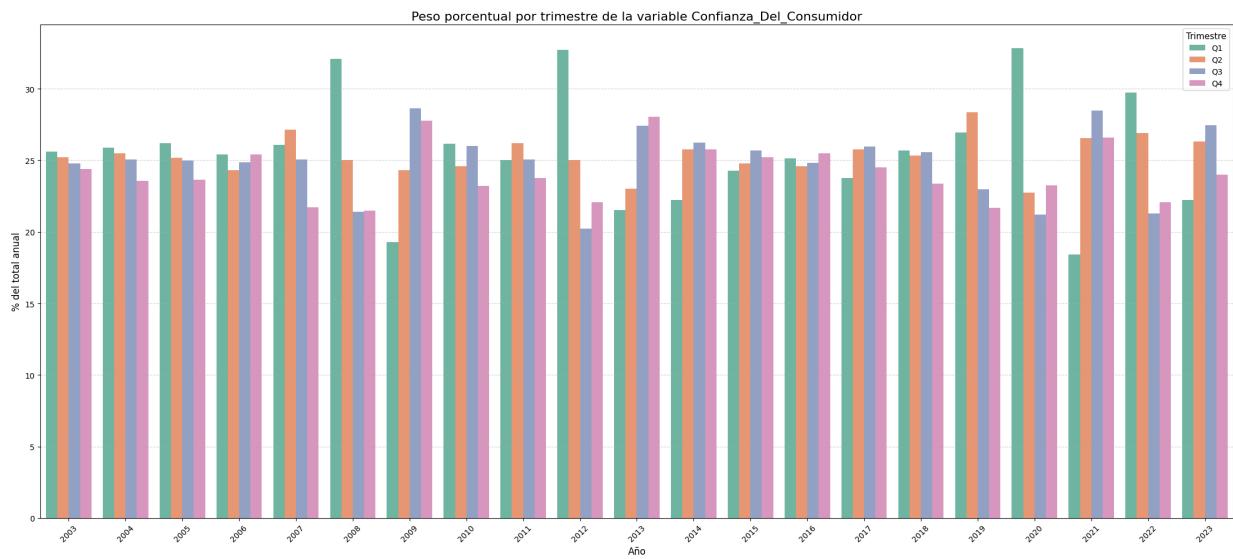
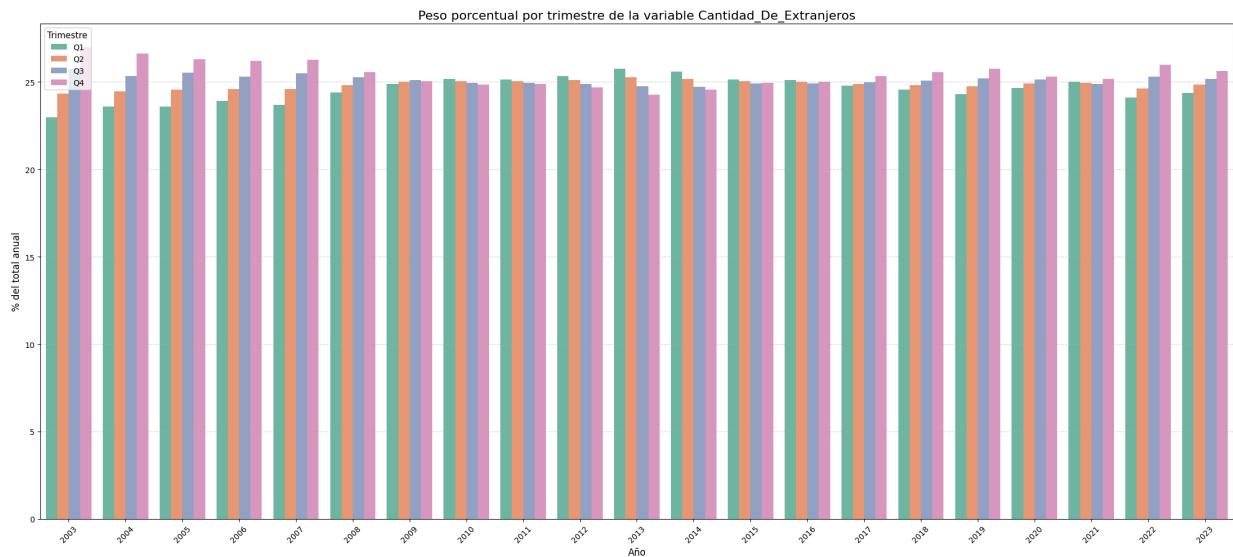
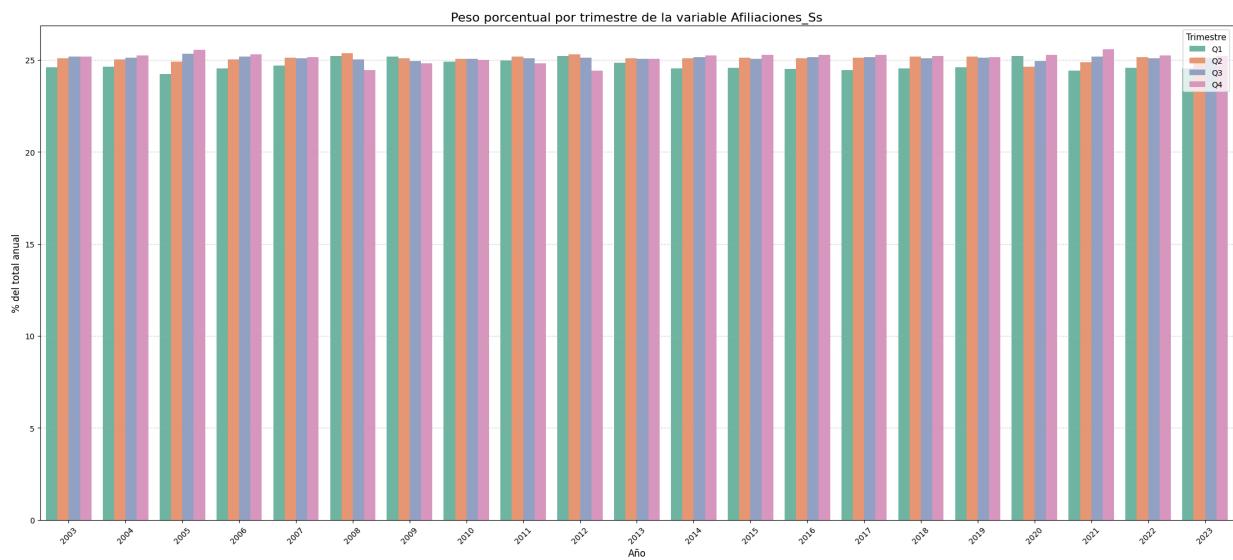
```
In [20]: df_clean['Año'] = df_clean['Fecha'].dt.year
df_clean['Trimestre_simple'] = df_clean['Fecha'].dt.quarter.map({1: 'Q1', 2: 'Q2', 3: 'Q3', 4: 'Q4'})

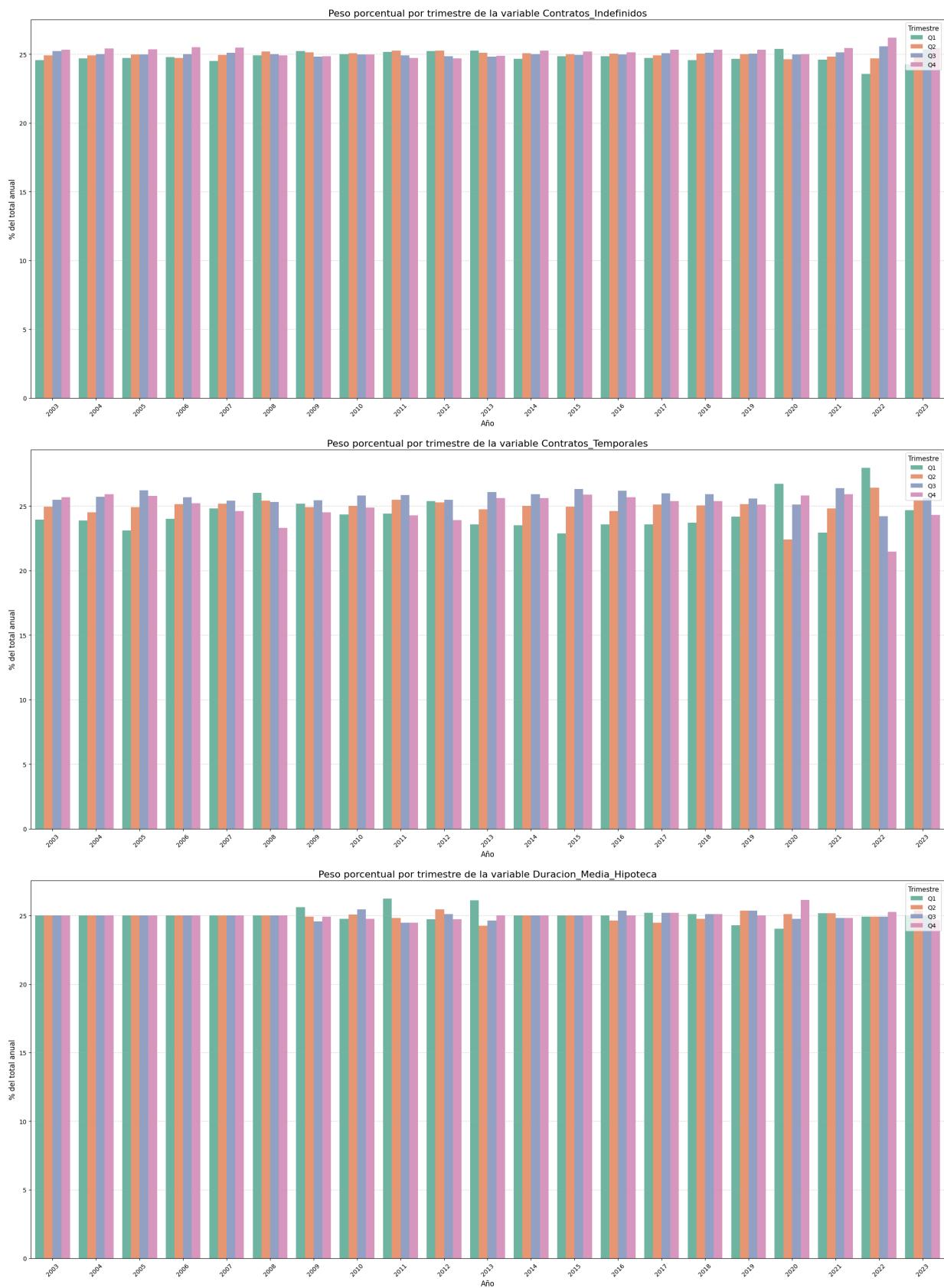
cols = df_clean.select_dtypes(include=['float64', 'int64']).columns.difference(['Fecha_ordinal'])

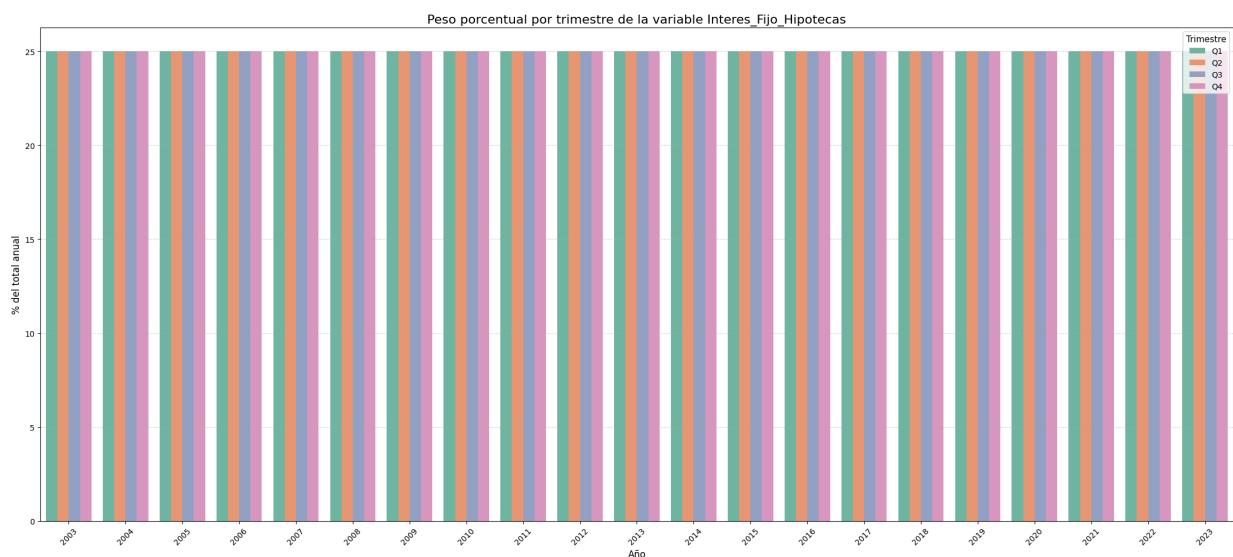
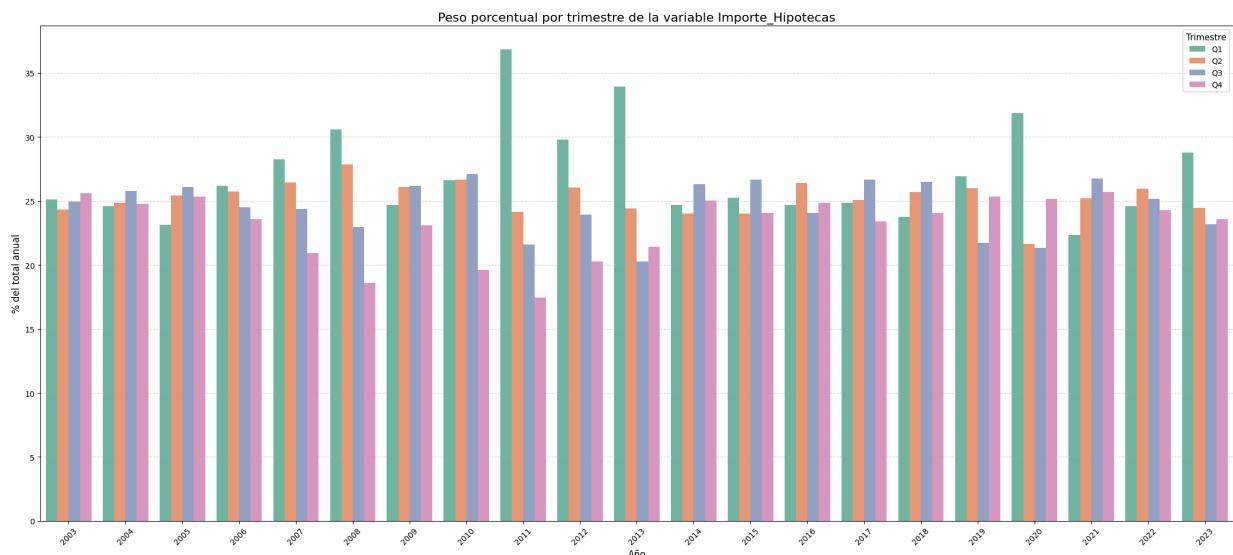
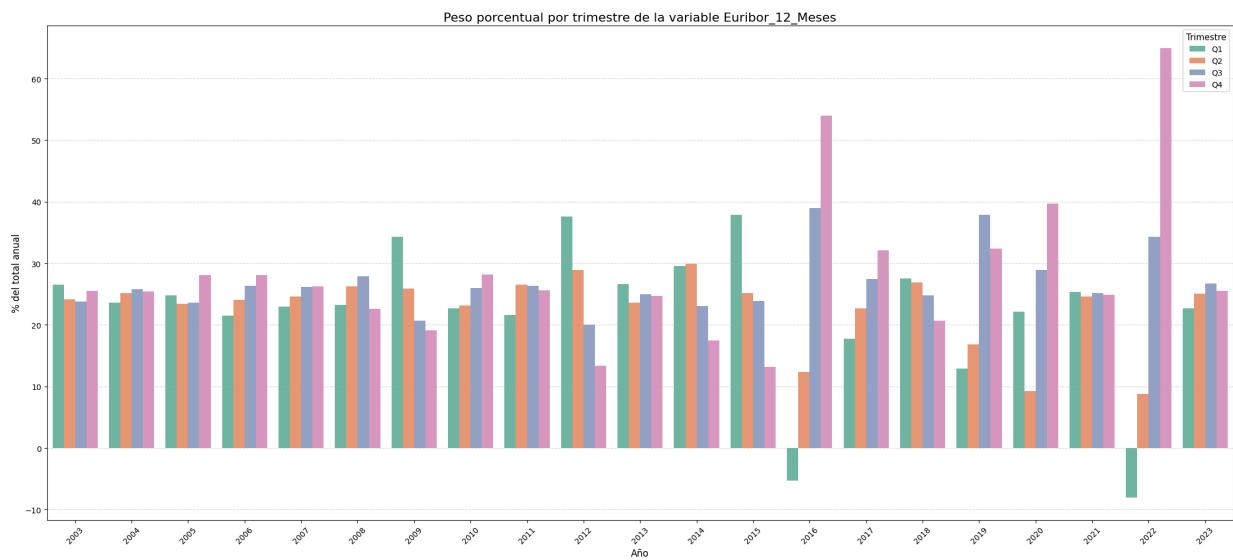
for var in cols:
    df_peso = df_clean.groupby(['Año', 'Trimestre_simple'])[var].sum().reset_index()
    df_peso['Total_Annual'] = df_peso.groupby('Año')[var].transform('sum')
    df_peso['Proporción'] = df_peso[var] / df_peso['Total_Annual'] * 100

    plt.figure(figsize=(22, 10)) # Gráfico más ancho y alto
    sns.barplot(data=df_peso, x='Año', y='Proporción', hue='Trimestre_simple', palette='Set2')

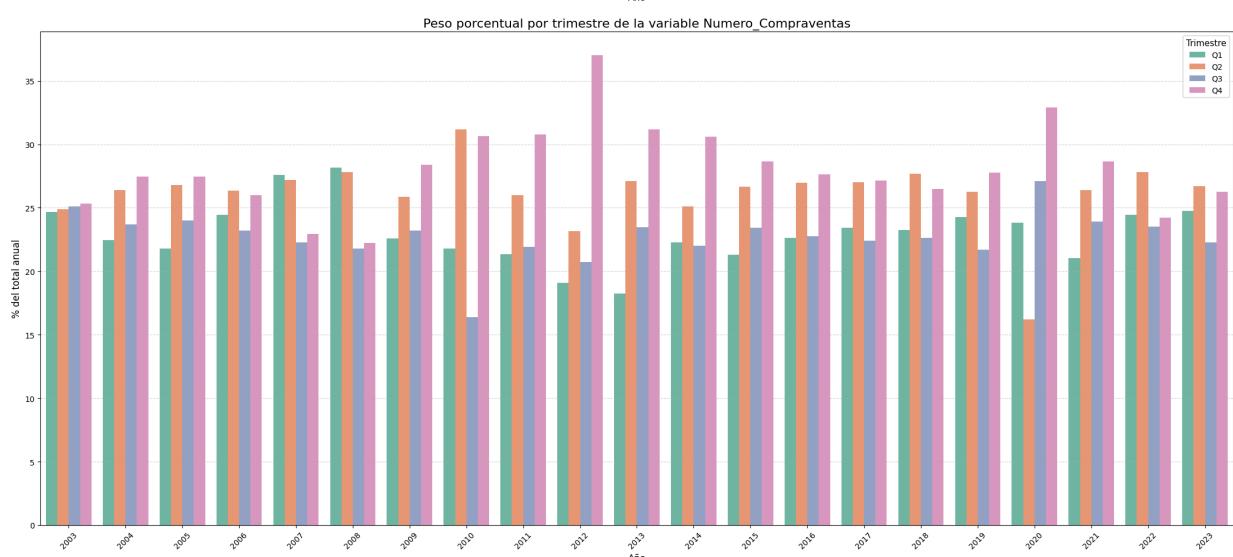
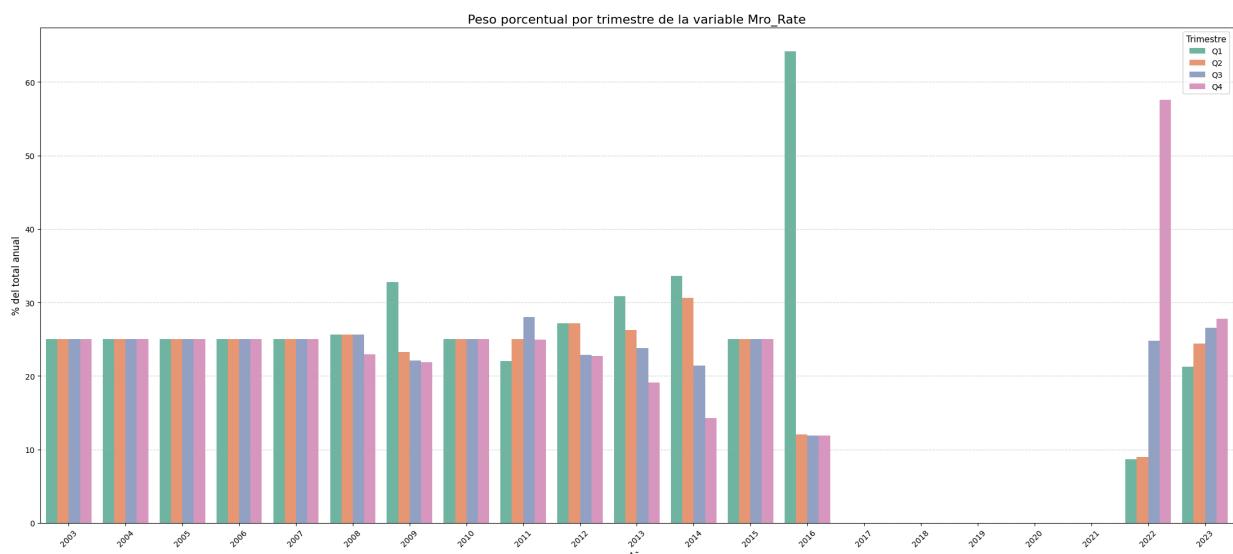
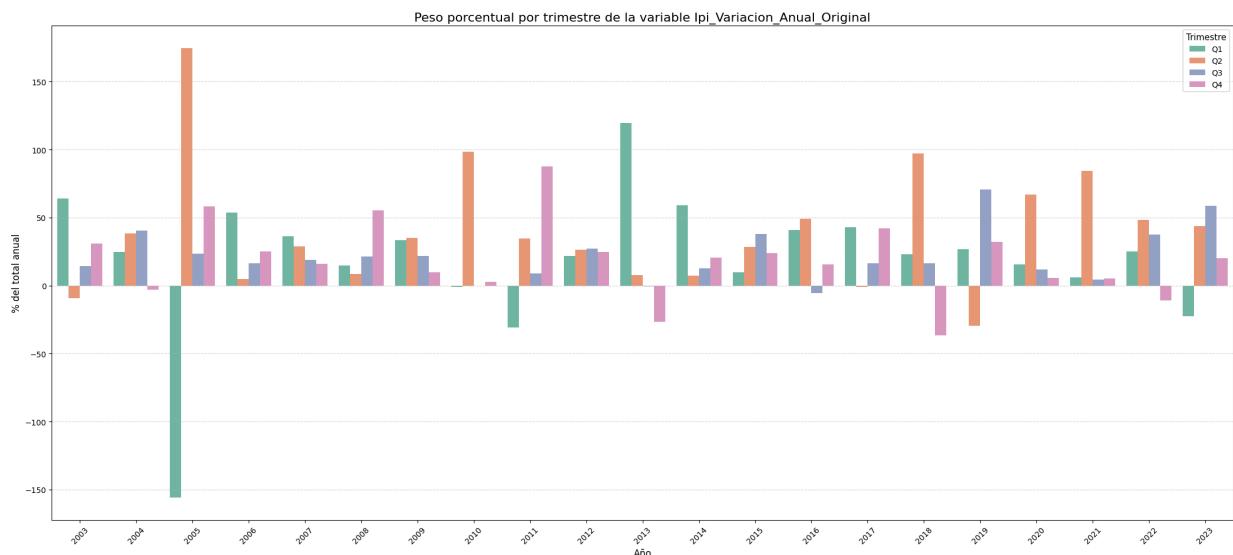
    plt.title(f'Peso porcentual por trimestre de la variable {var}', fontsize=16)
    plt.ylabel('% del total anual', fontsize=12)
    plt.xlabel('Año', fontsize=12)
    plt.xticks(rotation=45, fontsize=10)
    plt.yticks(fontsize=10)
    plt.grid(True, axis='y', linestyle='--', alpha=0.5)
    plt.legend(title='Trimestre', fontsize=10, title_fontsize=11)
    plt.tight_layout()
    plt.show()
```

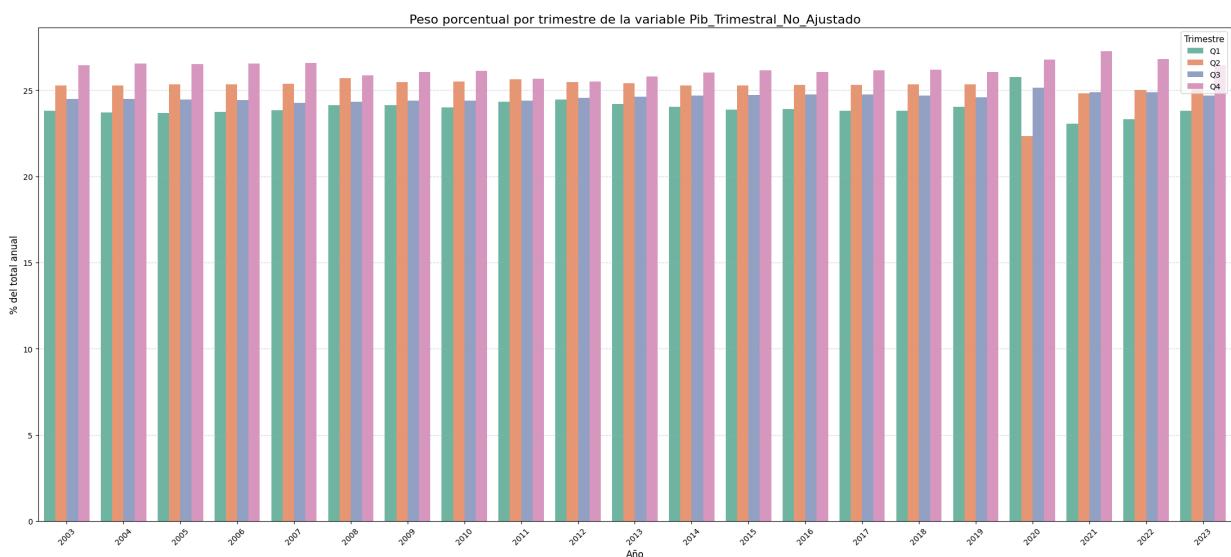
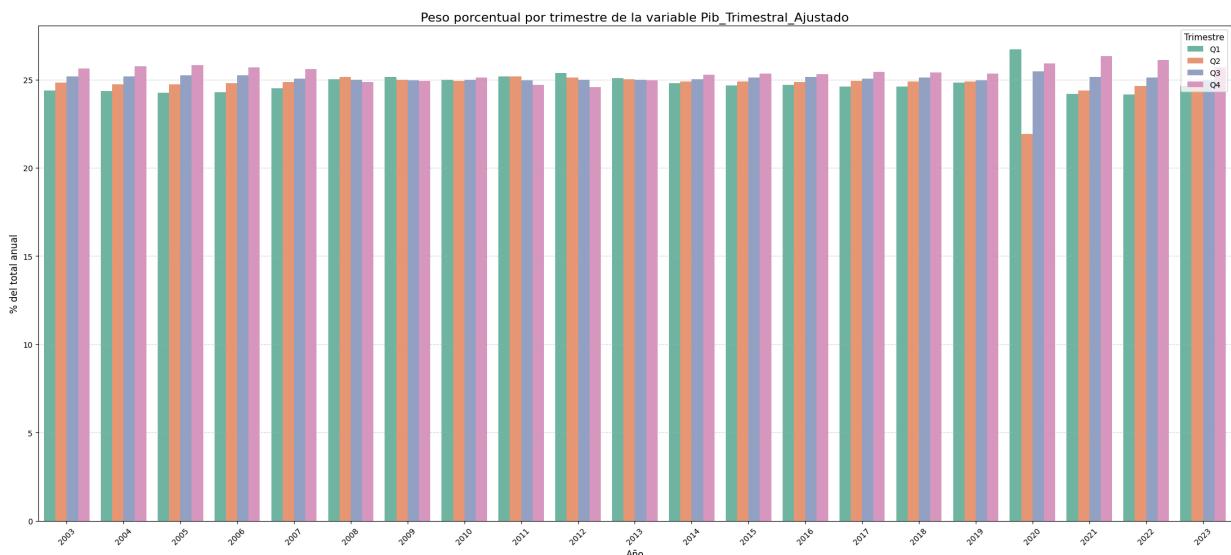
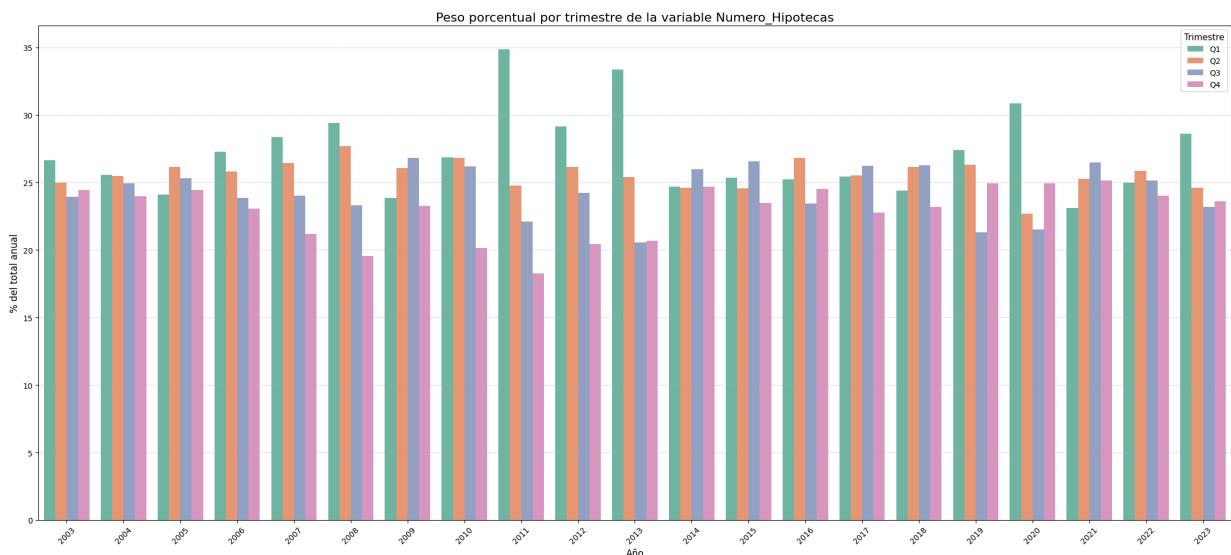


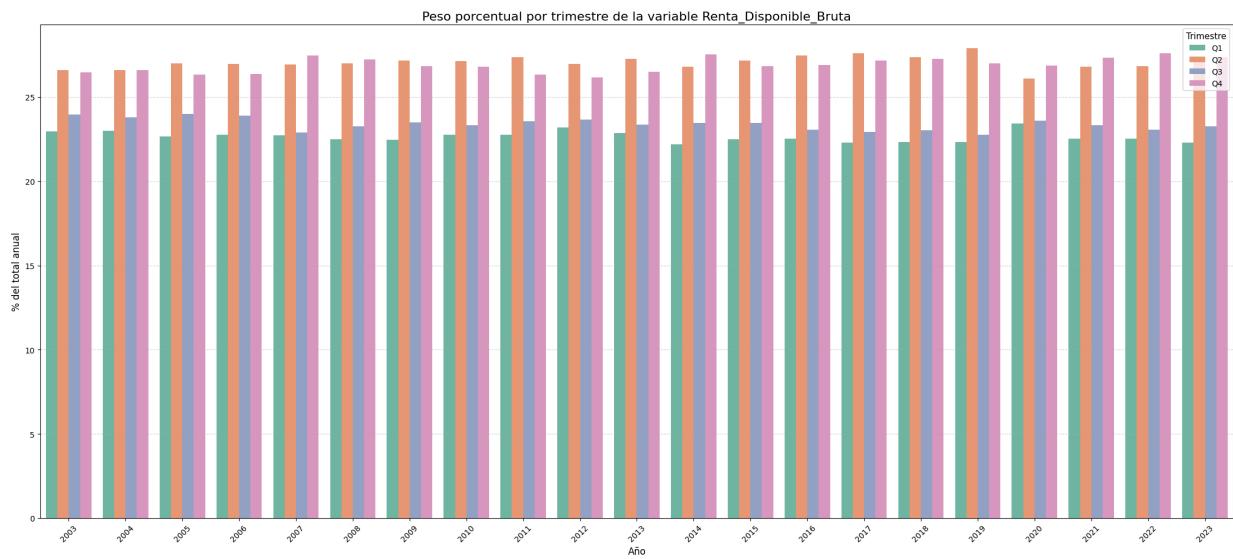
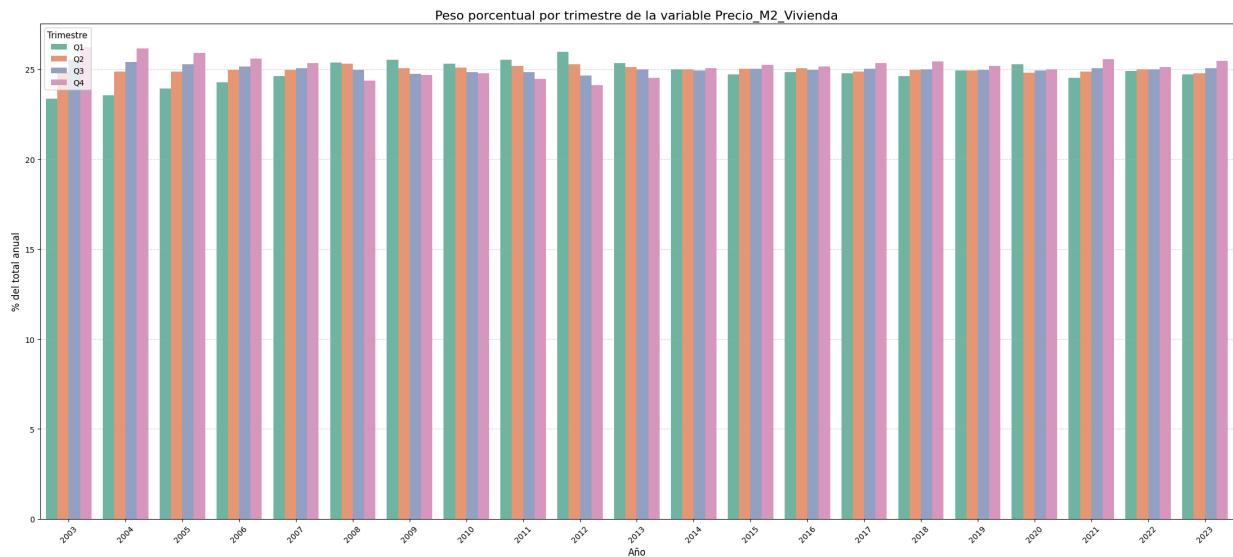
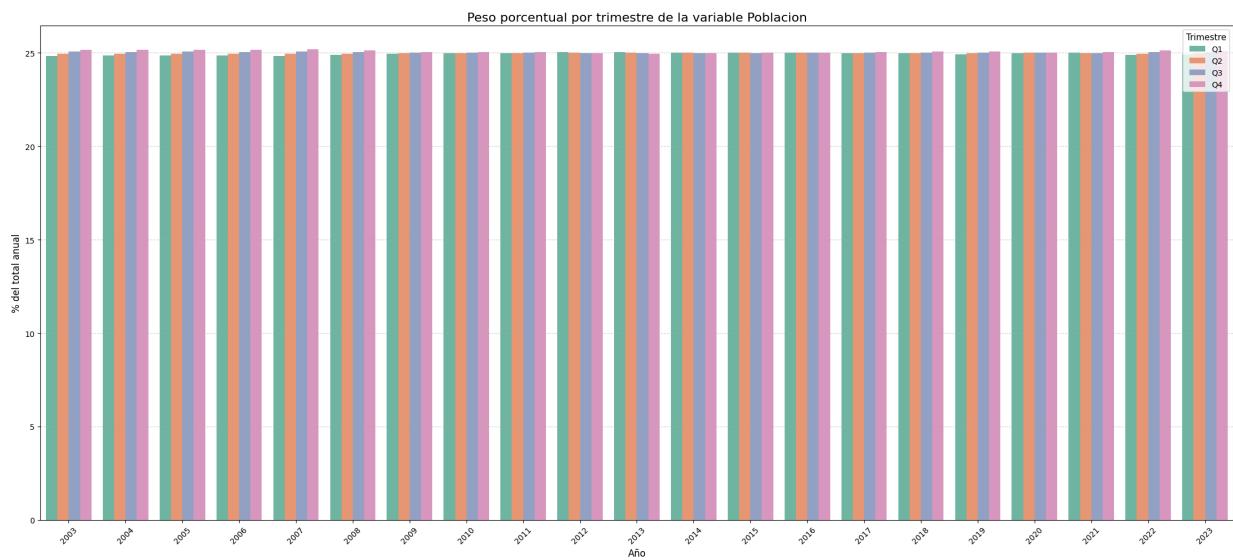


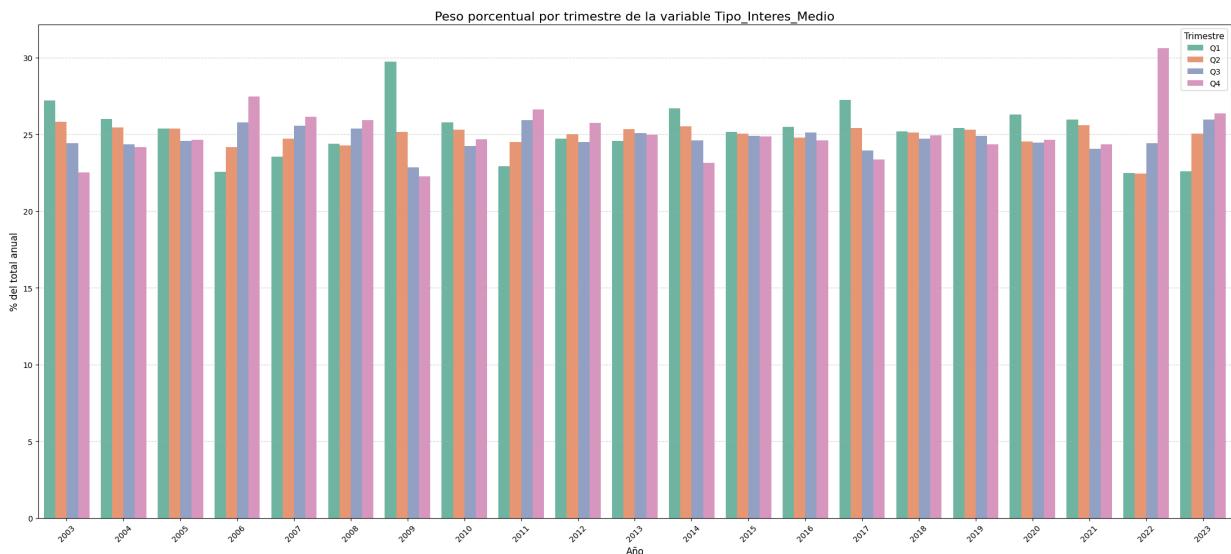
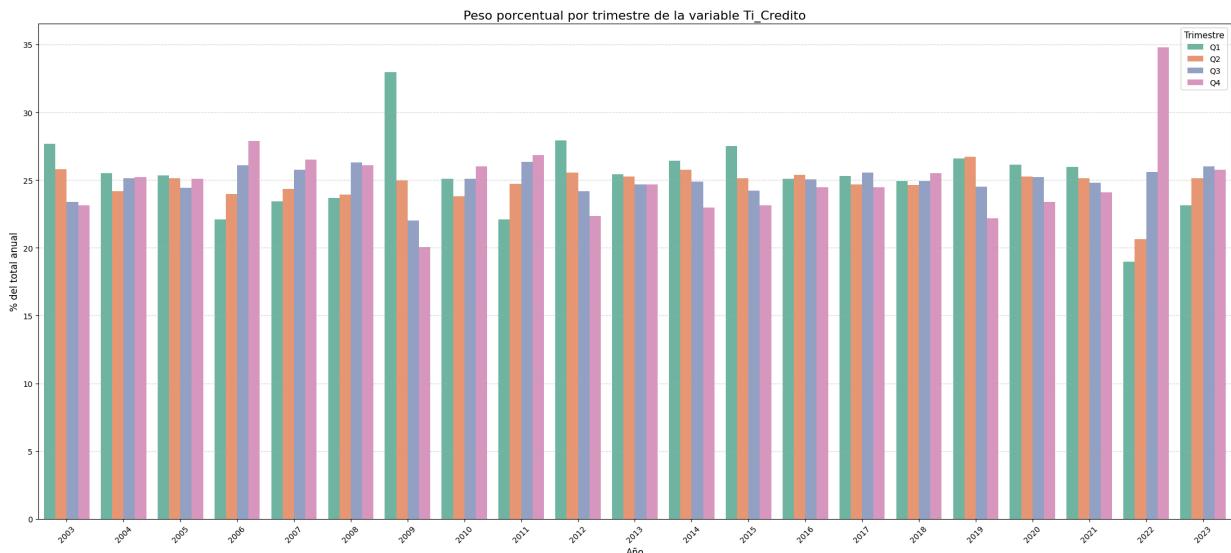
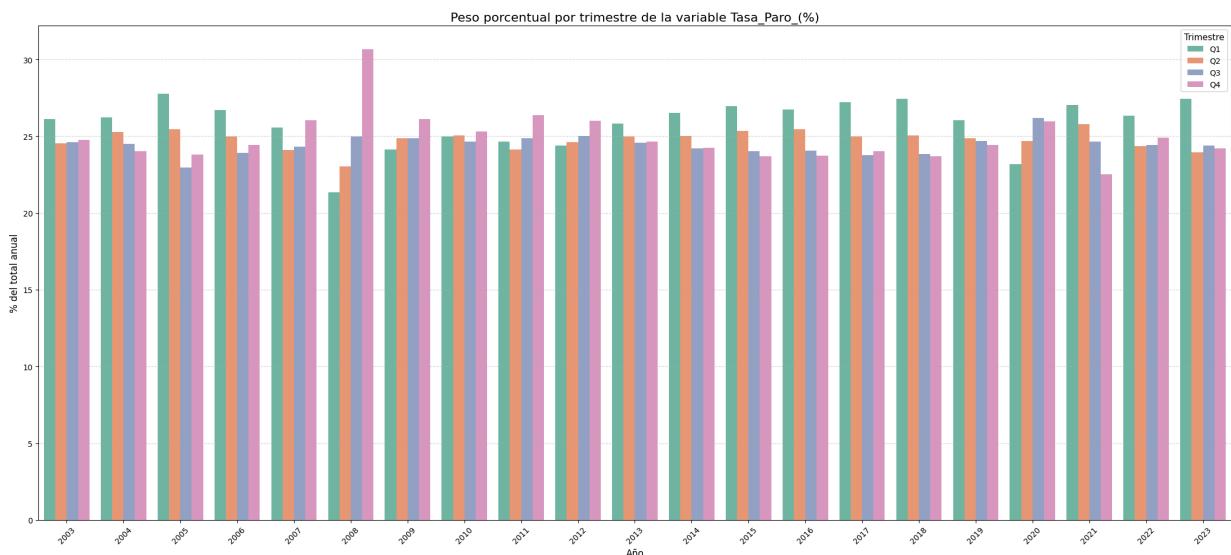


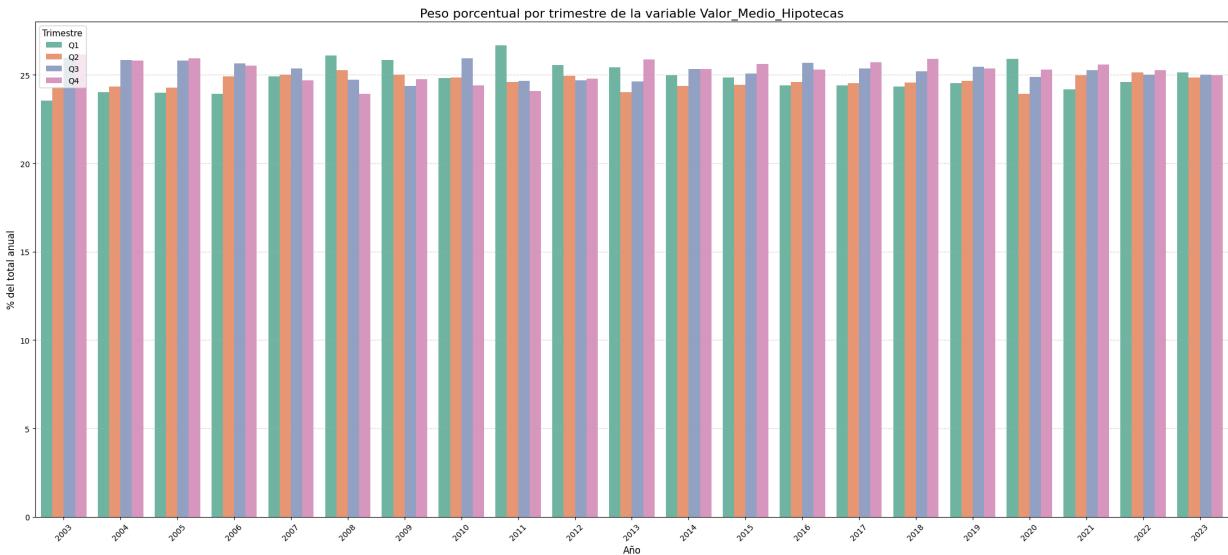
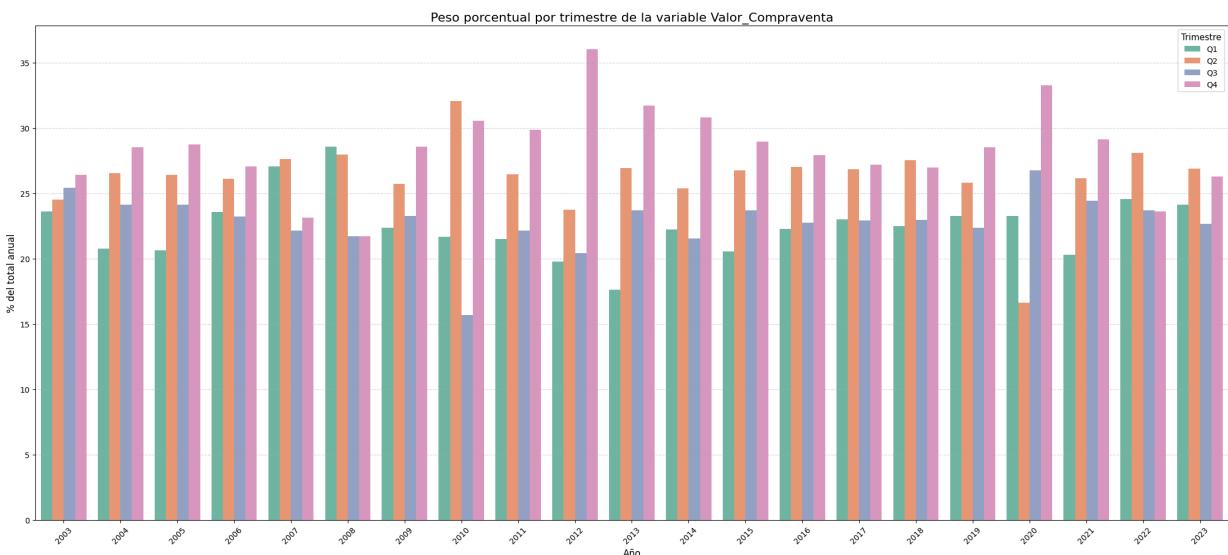












4. Análisis de outliers

Detección de outliers con Z-score

El Z-score es un método apropiado para identificar como outliers los valores con $|Z| > 3$; es decir, más de 3 desviaciones estándar de la media. De esta manera, podremos conocer cuántos outliers hay en cada variable. No obstante, es importante mencionar que es útil cuando las variables tienen una distribución normal o aproximadamente normal.

```
In [21]: num_vars = df_clean.select_dtypes(include=['float64', 'int64']).columns.difference(['Fecha_ordinal'])

print("\n📌 Outliers detectados con Z-score (|z| > 3):")
outliers_z = {}
for col in num_vars:
    z_scores = zscore(df_clean[col].dropna())
    outliers = np.where(np.abs(z_scores) > 3)[0]
    if len(outliers) > 0:
        outliers_z[col] = len(outliers)
        print(f'{col}: {len(outliers)} valores atípicos')

📌 Outliers detectados con Z-score (|z| > 3):
Contratos_Definidos: 2 valores atípicos
Ipc_Variación_Anual: 2 valores atípicos
Ipi_Variacion_Anual_Corregida: 3 valores atípicos
Ipi_Variacion_Anual_Original: 4 valores atípicos
Pib_Trimestral_No_Ajustado: 1 valores atípicos
```

Detección de outliers con IQR

En este caso, usamos el método del rango intercuartílico (IQR) para detectar valores atípicos, que consiste en detectar como outliers cualquier valor fuera de $[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$. Esta sí es una técnica robusta a la distribución, por lo que permite identificar correctamente outliers en variables sesgadas o que no siguen una distribución normal.

```
In [22]: print("\n✿ Outliers detectados con IQR:")
outliers_iqr = {}
for col in num_vars:
    Q1 = df_clean[col].quantile(0.25)
    Q3 = df_clean[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df_clean[(df_clean[col] < lower_bound) | (df_clean[col] > upper_bound)]
    if not outliers.empty:
        outliers_iqr[col] = outliers.shape[0]
print(f"\n{col}: {outliers_iqr[col]} valores atípicos")
```

✿ Outliers detectados con IQR:
 Cantidad_De_Extranjeros: 6 valores atípicos
 Contratos_Indefinidos: 6 valores atípicos
 Duracion_Media_Hipoteca: 12 valores atípicos
 Importe_Hipotecas: 4 valores atípicos
 Ipc_%_Variación_Anual: 3 valores atípicos
 Ipi_Variacion_Anual_Corregida: 6 valores atípicos
 Ipi_Variacion_Anual_Original: 6 valores atípicos
 Pib_Trimestral_Ajustado: 6 valores atípicos
 Pib_Trimestral_No_Ajustado: 7 valores atípicos
 Poblacion: 15 valores atípicos
 Renta_Disponible_Bruta: 2 valores atípicos
 Ti_Credito: 2 valores atípicos

Como podemos comprobar, obtenemos más outliers con el método IQR, ya que como hemos comentado es más robusto frente a la forma de la distribución (especialmente en distribuciones sesgadas o con colas más largas). Por esta razón, el método IQR suele detectar más outliers que Z-score. En nuestro caso, dado que estamos trabajando con datos económicos que no suelen tener una distribución normal (como comprobamos anteriormente), el método IQR será generalmente más útil y fiable

Detección de variables con alta asimetría (skew)

A continuación, vamos a calcular el coeficiente de asimetría (skewness) para cada variable, que nos va a ayudar a identificar las variables que tienen una alta asimetría ($|\text{skew}| > 1$), pues estas pueden dificultar modelos lineales. Esto es así porque variables muy asimétricas violan el supuesto de normalidad de errores en modelos económicos. Además, vamos a aplicar distintas transformaciones a estas variables para ver si así conseguimos mejorar la kurtosis:

- **log1p**: ayuda a reducir la asimetría positiva (distribuciones sesgadas hacia la derecha) y a comprimir grandes valores para hacer la distribución más cercana a la normal. Es la función $\log(1 + x)$, donde \log es el logaritmo natural, y se usa en lugar del logaritmo simple para evitar problemas cuando x es cero (porque $\log(0)$ no está definido).
- **sqrt**: es simplemente la raíz cuadrada de los valores de la variable y también se usa para reducir la asimetría positiva, pero suele ser menos agresiva que el logaritmo. Funciona bien para variables con distribución sesgada, especialmente cuando los valores son todos positivos y no demasiado grandes. El objetivo de estas transformaciones es acercar la distribución de las variables a la normal, ya que muchas técnicas estadísticas (como la regresión lineal que realizaremos más adelante) asumen que las variables siguen una distribución aproximadamente normal.

```
In [23]: num_vars = df_clean.select_dtypes(include=[np.number]).columns.difference(['Fecha_ordinal', 'Número_Hipotecas'])

skewed = df_clean[num_vars].apply(skew).sort_values(ascending=False)

threshold = 1
high_skew_vars = skewed[abs(skewed) > threshold].index.tolist()

print("✿ Variables con skew > 1 (candidatas a transformar):")
print(high_skew_vars)

for col in high_skew_vars:
    fig, axs = plt.subplots(1, 3, figsize=(15, 4))

    sns.histplot(df_clean[col], kde=True, ax=axs[0], color='skyblue')
    axs[0].set_title(f"\n{col} - Original (skew={df_clean[col].skew():.2f})")

    log_trans = np.log1p(df_clean[col])
    sns.histplot(log_trans, kde=True, ax=axs[1], color='orange')
    axs[1].set_title(f"\n{col} - Log1p (skew={log_trans.skew():.2f})")
```

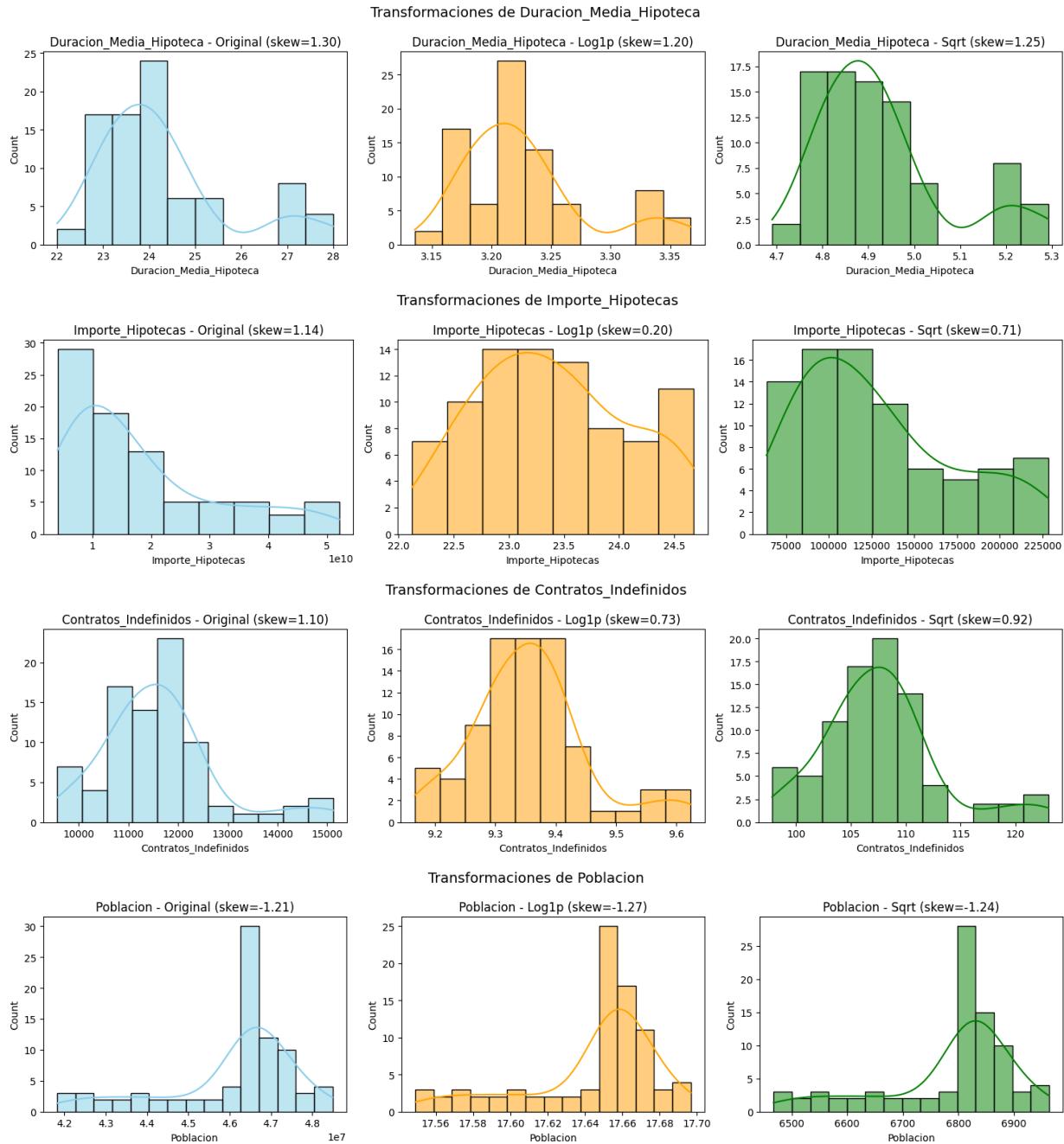
```

sqrt_trans = np.sqrt(df_clean[col])
sns.histplot(sqrt_trans, kde=True, ax= axs[2], color='green')
axs[2].set_title(f" {col} - Sqrt (skew={sqrt_trans.skew():.2f})")

plt.suptitle(f'Transformaciones de {col}', fontsize=14)
plt.tight_layout()
plt.show()

```

❖ Variables con skew > 1 (candidatas a transformar):
['Duracion_Media_Hipoteca', 'Importe_Hipotecas', 'Contratos_Indefinidos', 'Poblacion']



Una vez que hemos visto las transformaciones gráficamente, creamos un nuevo dataframe con estas variables transformadas a través del método que mejore su kurtosis (es decir, donde sea más cercana a cero).

```

In [24]: df_transformed = df_clean.copy()

threshold = 1

num_vars = df_clean.select_dtypes(include=[np.number]).columns.difference(['Fecha_ordinal', 'Numero_Hipotecas'])

skewed = df_clean[num_vars].apply(lambda x: x.skew()).sort_values(ascending=False)
high_skew_vars = skewed[abs(skewed) > threshold].index.tolist()

for col in high_skew_vars:
    original_skew = df_clean[col].skew()

```

```

log_trans = np.log1p(df_clean[col])
sqrt_trans = np.sqrt(df_clean[col])

skew_log = log_trans.skew()
skew_sqrt = sqrt_trans.skew()

options = {
    'original': (df_clean[col], abs(original_skew)),
    'log1p': (log_trans, abs(skew_log)),
    'sqrt': (sqrt_trans, abs(skew_sqrt))
}

best_transform = min(options.items(), key=lambda x: x[1][1])

if best_transform[1][1] < abs(original_skew):
    df_transformed[col] = best_transform[1][0]
    print(f"Variable '{col}' transformada con '{best_transform[0]}' (skew original: {original_skew:.2f} -> {best_transform[1][1]})")
else:
    print(f"Variable '{col}' no se transforma (skew original: {original_skew:.2f})")

```

Variable 'Duracion_Media_Hipoteca' transformada con 'log1p' (skew original: 1.30 -> skew transformada: 1.20)
 Variable 'Importe_Hipotecas' transformada con 'log1p' (skew original: 1.14 -> skew transformada: 0.20)
 Variable 'Contratos_Indefinidos' transformada con 'log1p' (skew original: 1.10 -> skew transformada: 0.73)
 Variable 'Poblacion' no se transforma (skew original: -1.21)

In [25]: `df_transformed.head()`

Out[25]:

	Fecha	Numero_Hipotecas	Importe_Hipotecas	Valor_Medio_Hipotecas	Tipo_Interes_Medio	Tasa_Paro_(%)	Contratos_Inde
0	2003-03-31	263600	23.908609	91713.17	4.693333	11.99	9.
1	2003-06-30	247071	23.876021	94711.40	4.453333	11.28	9.
2	2003-09-30	236977	23.900416	101184.20	4.213333	11.30	9.
3	2003-12-31	241791	23.926874	101828.44	3.883333	11.37	9.
4	2004-03-31	283170	24.125194	106020.83	3.760000	11.50	9.

5 rows × 31 columns

Visualización de outliers

Es un complemento visual para la detección de outliers, basado en el método del rango intercuartílico (IQR)

In [26]:

```

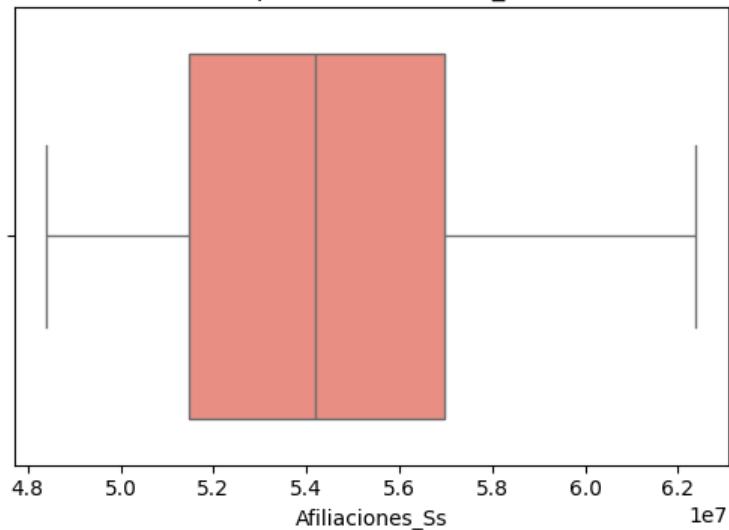
for col in num_vars:
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 2)
    sns.boxplot(x=df_clean[col], color='salmon')
    plt.title(f'Boxplot de {col}')

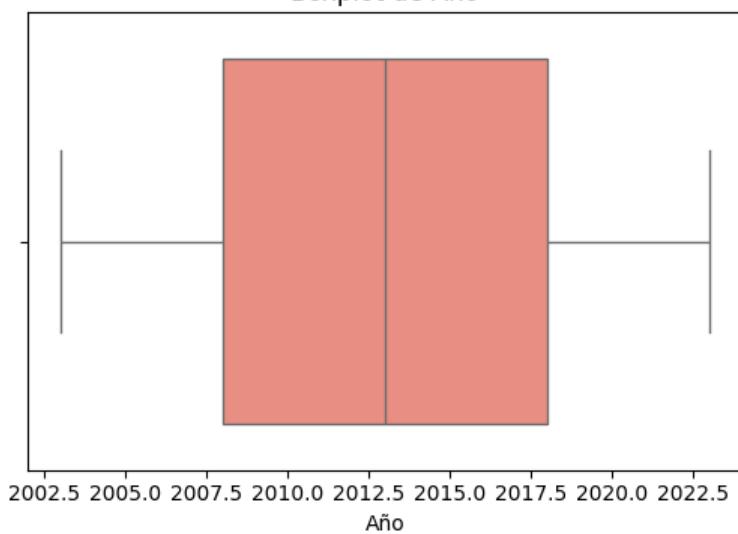
    plt.tight_layout()
    plt.show()

```

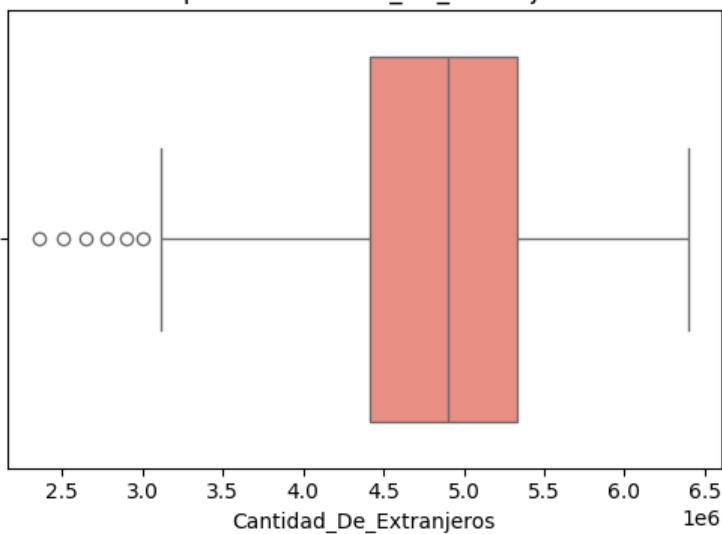
Boxplot de Afiliaciones_Ss



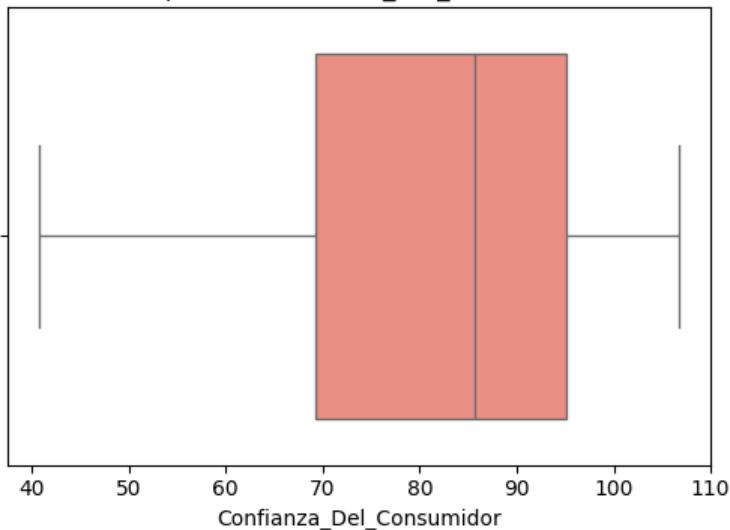
Boxplot de Año



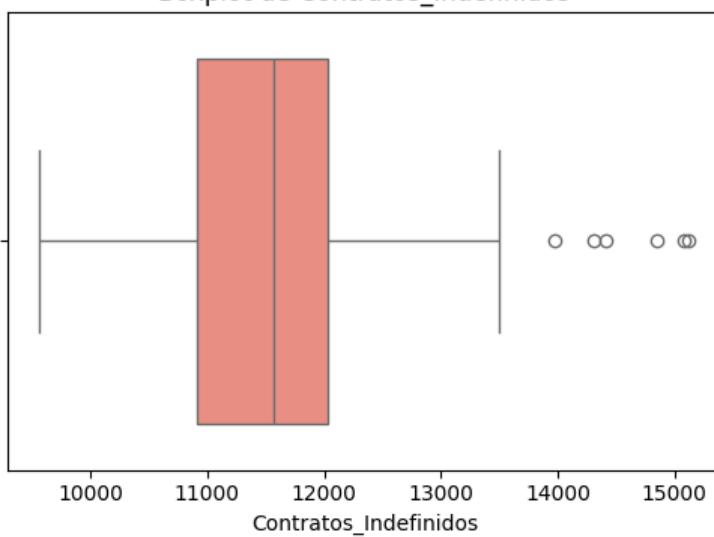
Boxplot de Cantidad_De_Extranjeros



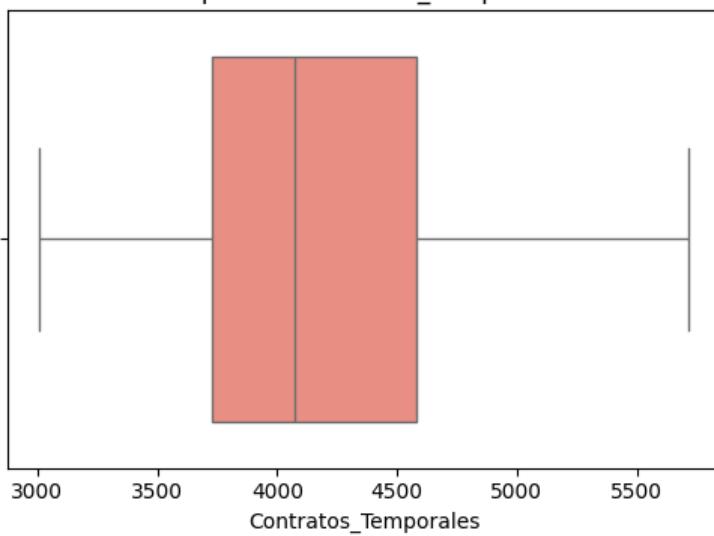
Boxplot de Confianza_Del_Consumidor



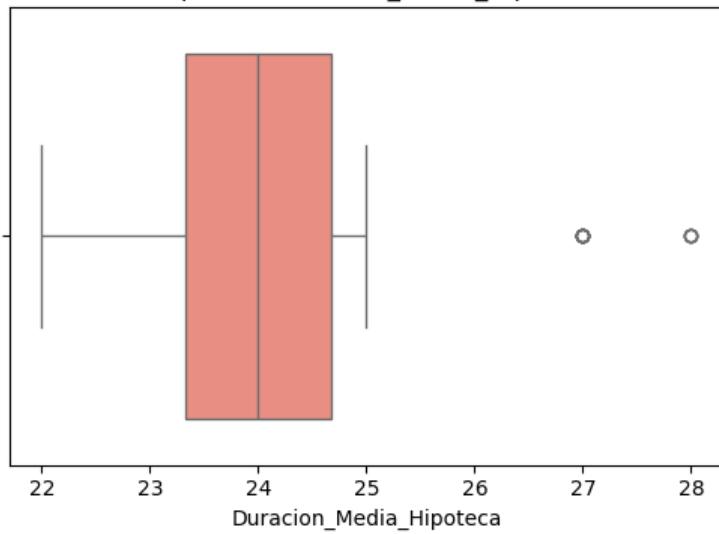
Boxplot de Contratos_Indefinidos



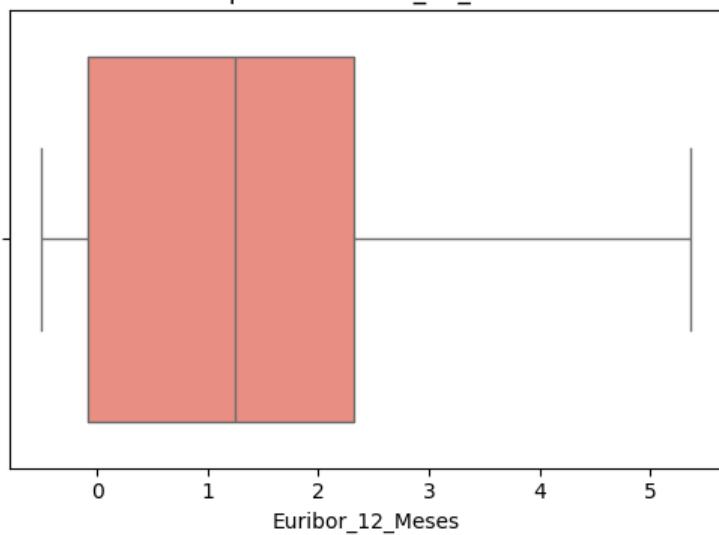
Boxplot de Contratos_Temporales



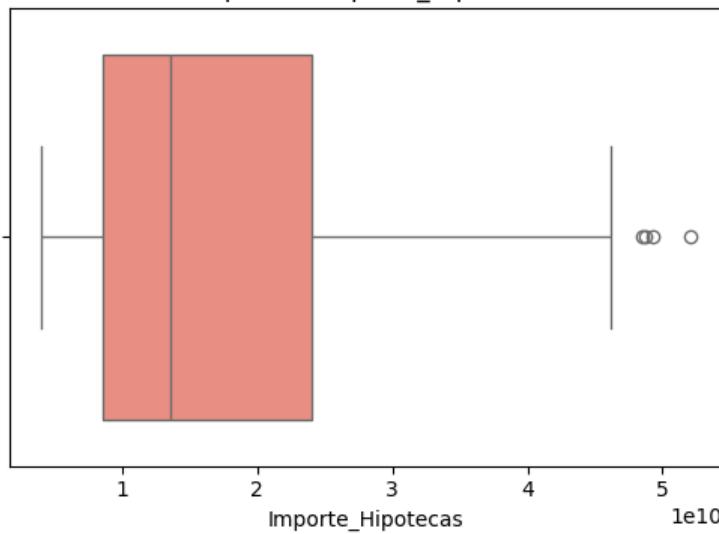
Boxplot de Duracion_Media_Hipoteca



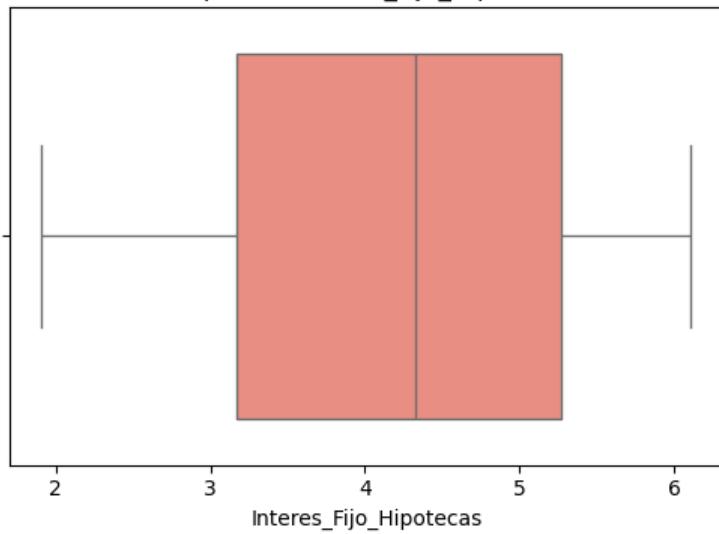
Boxplot de Euribor_12_Meses



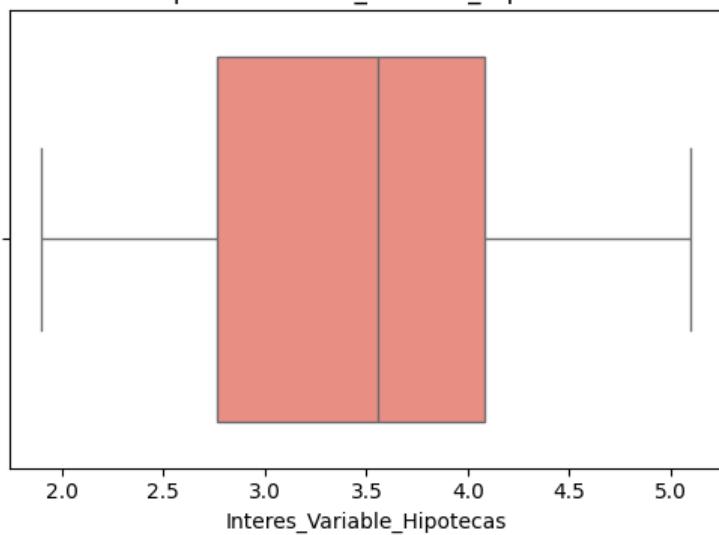
Boxplot de Importe_Hipotecas



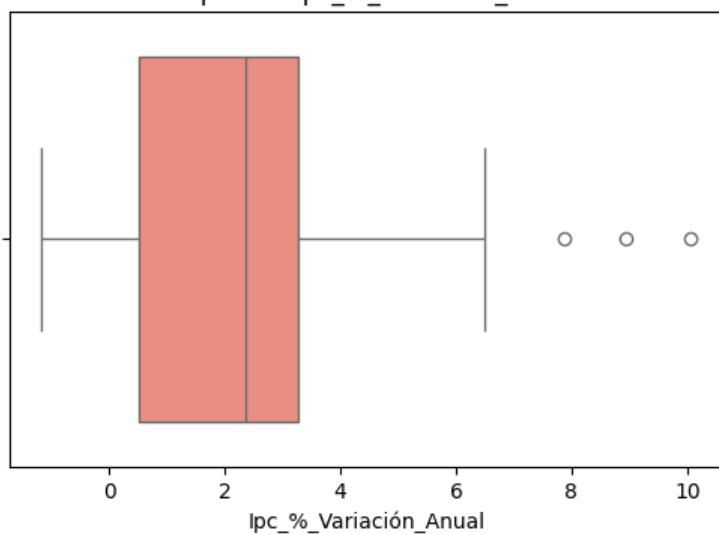
Boxplot de Interes_Fijo_Hipotecas



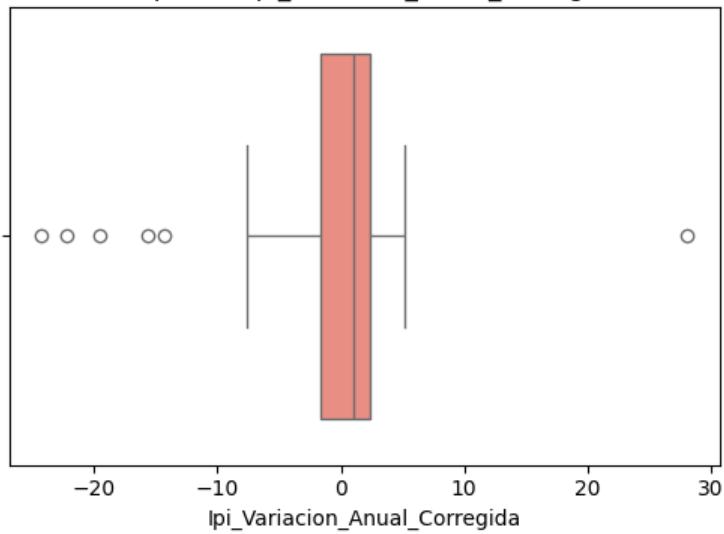
Boxplot de Interes_Variable_Hipotecas



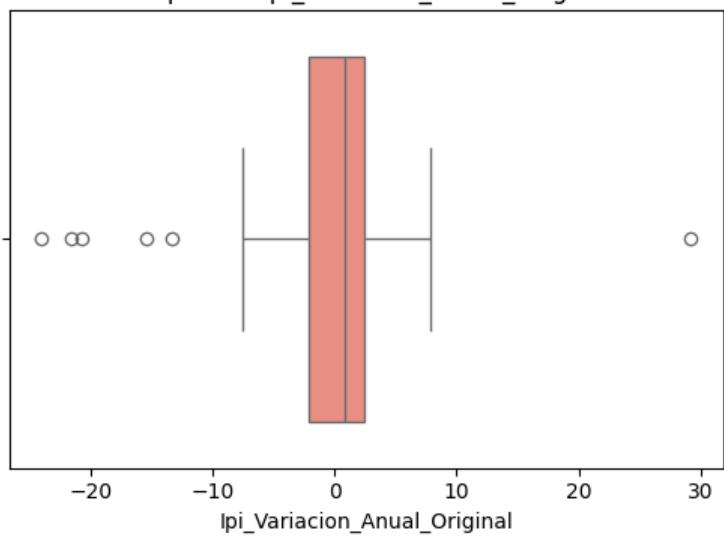
Boxplot de Ipc_%_Variación_Anual



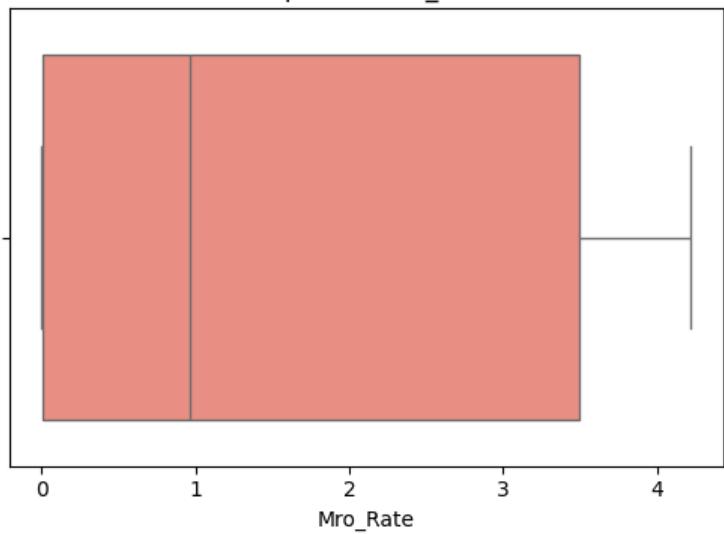
Boxplot de Ipi_Variacion_Anual_Corregida



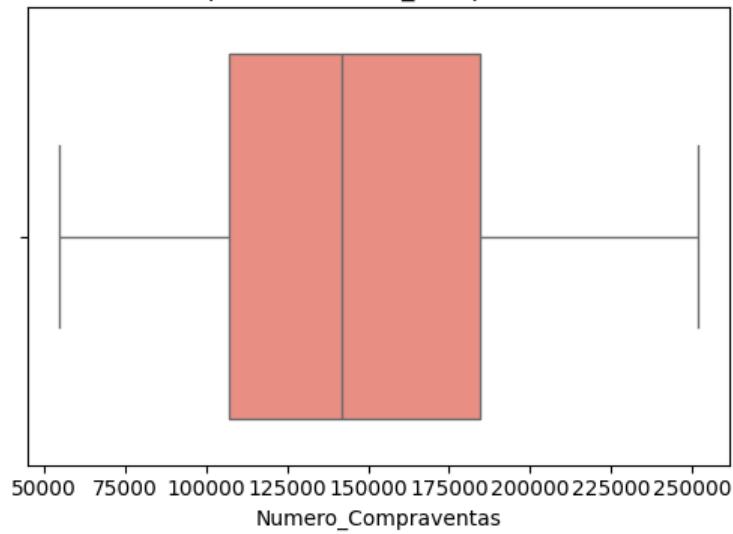
Boxplot de Ipi_Variacion_Anual_Original



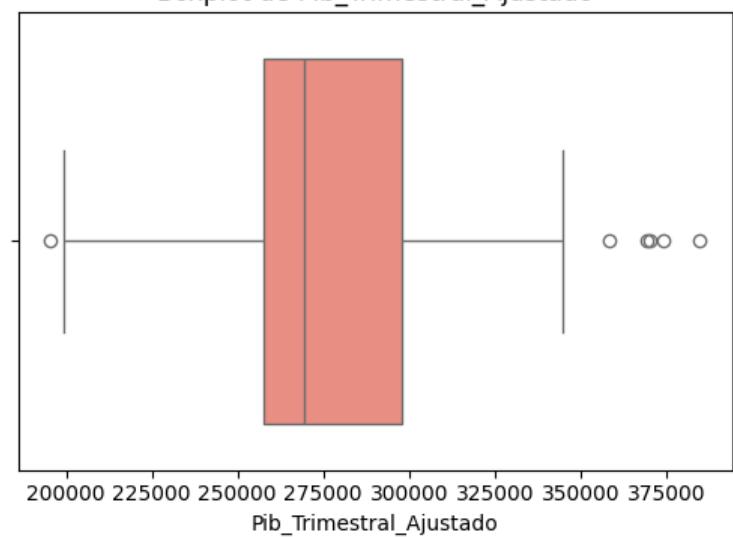
Boxplot de Mro_Rate



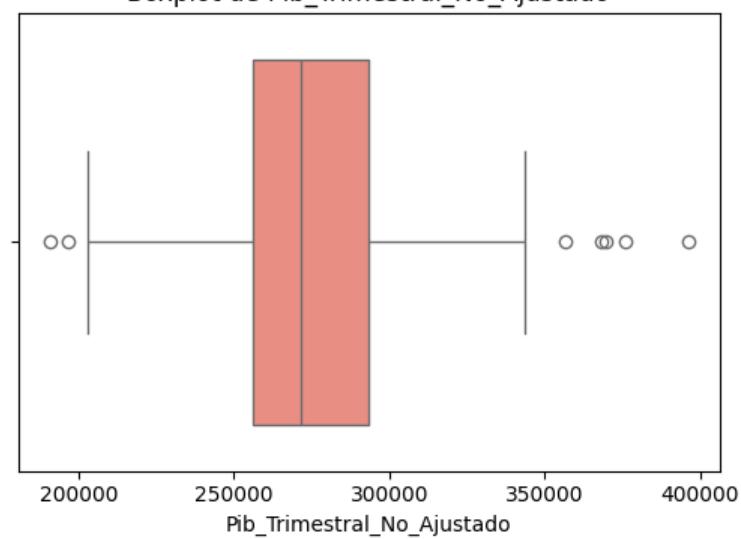
Boxplot de Numero_Compraventas



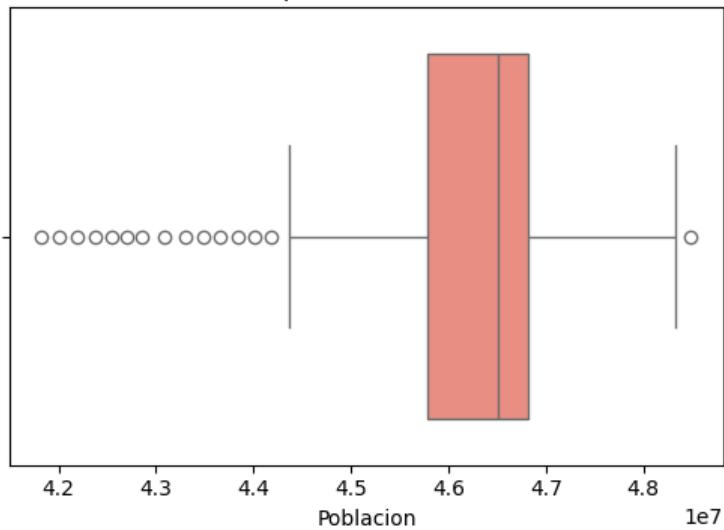
Boxplot de Pib_Trimestral_Ajustado



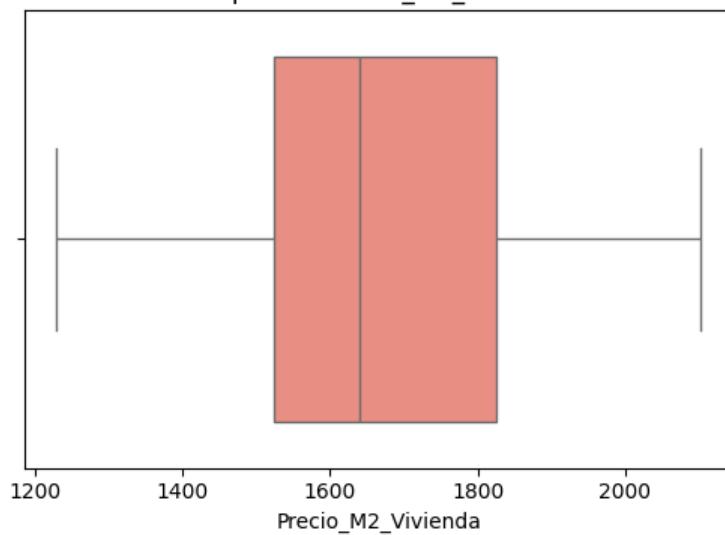
Boxplot de Pib_Trimestral_No_Ajustado



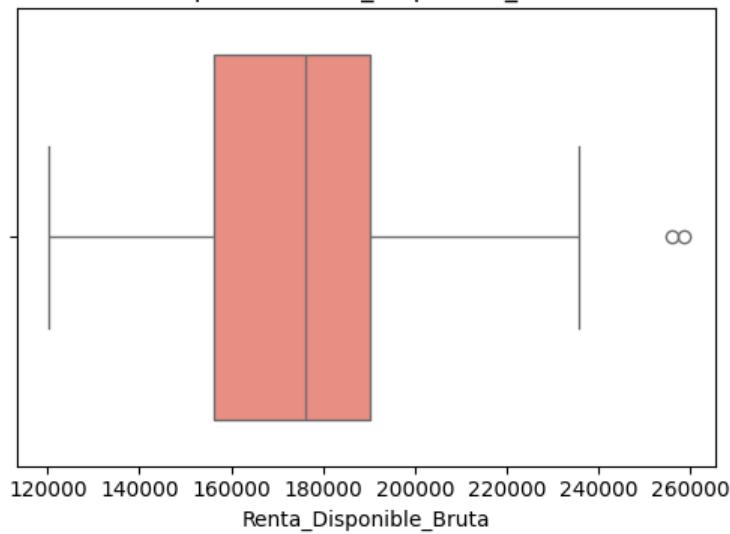
Boxplot de Poblacion



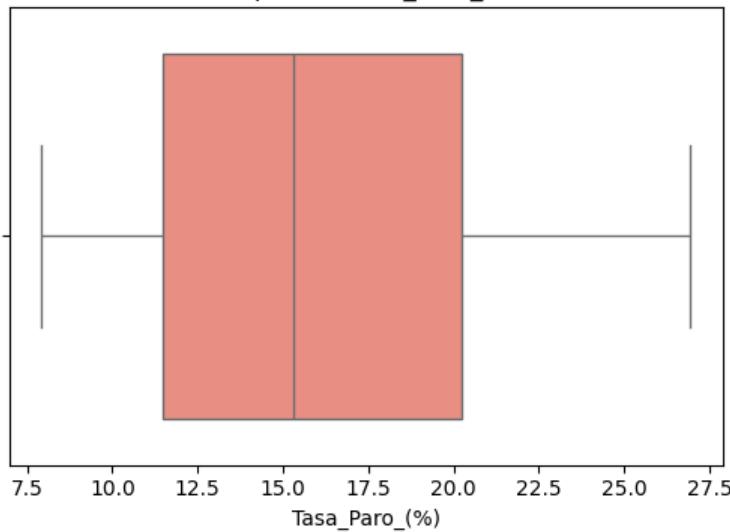
Boxplot de Precio_M2_Vivienda



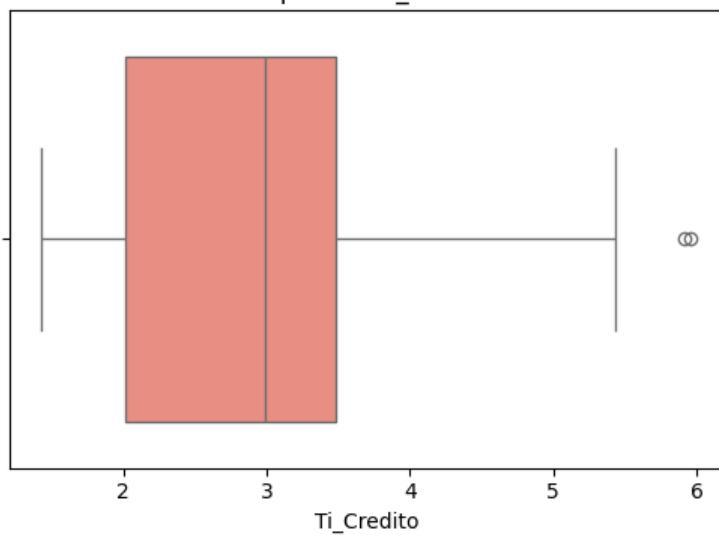
Boxplot de Renta_Disponible_Bruta



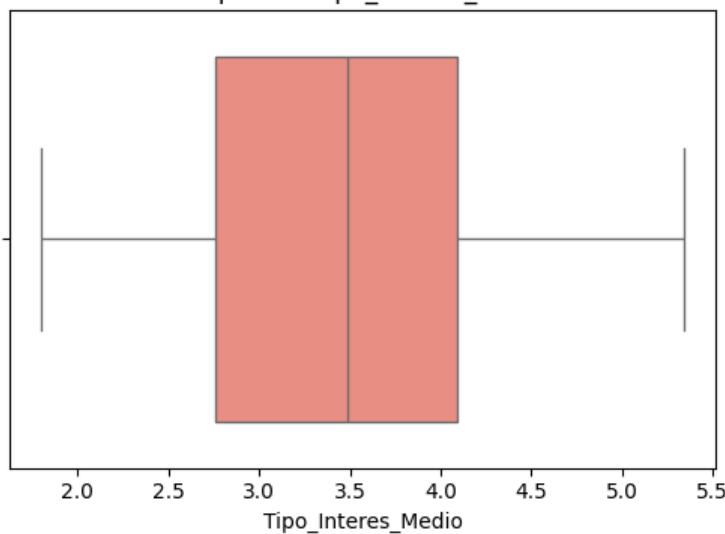
Boxplot de Tasa_Paro_(%)



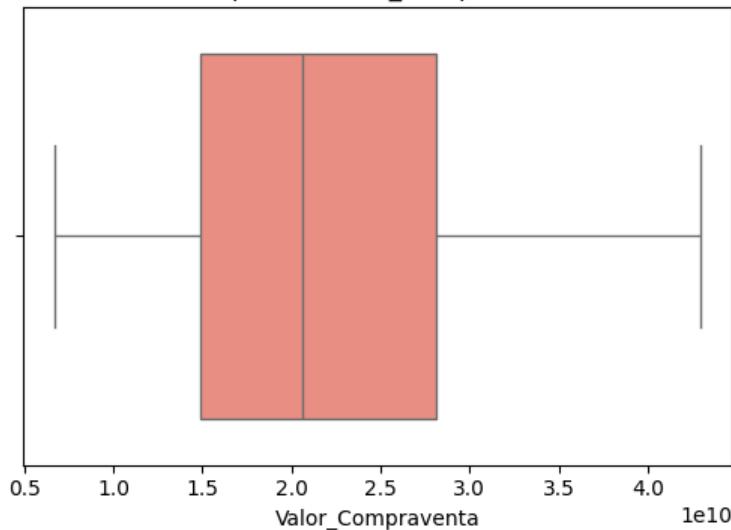
Boxplot de Ti_Credito



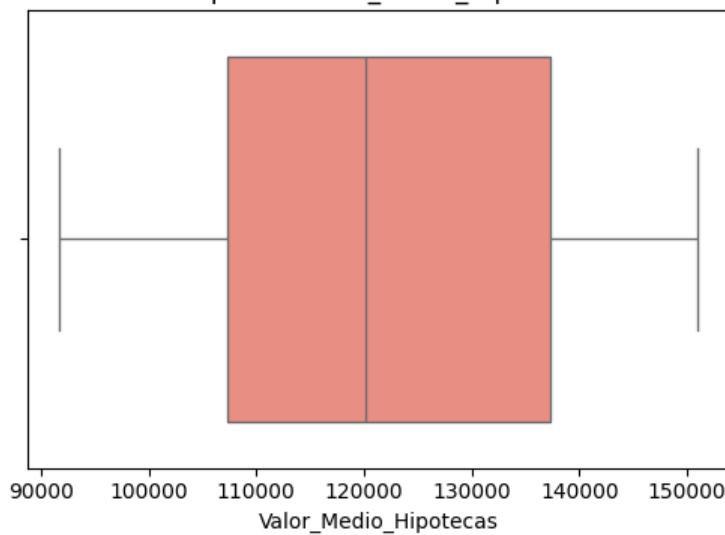
Boxplot de Tipo_Interes_Medio



Boxplot de Valor_Compraventa



Boxplot de Valor_Medio_Hipotecas



Comprobamos que las variables con outliers son las mismas que obteníamos en el apartado de Detección de outliers con IQR

No obstante, vamos a volver a calcular los outliers con IQR y gráficamente pero para el nuevo df_transformed que hemos obtenido tras las transformaciones de variables con kurtosis elevada

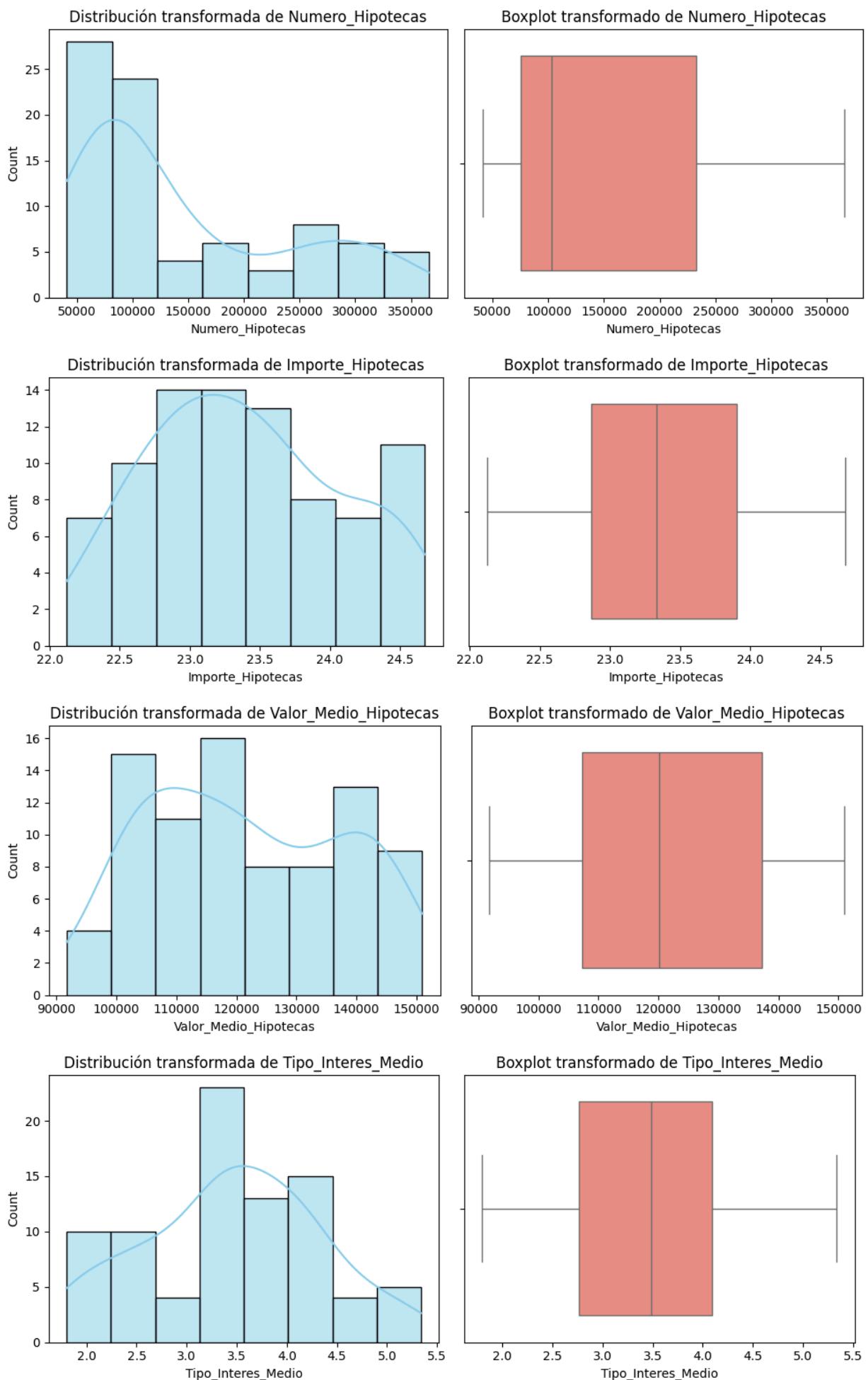
```
In [27]: num_vars = df_transformed.select_dtypes(include=['float64', 'int64']).columns.tolist()

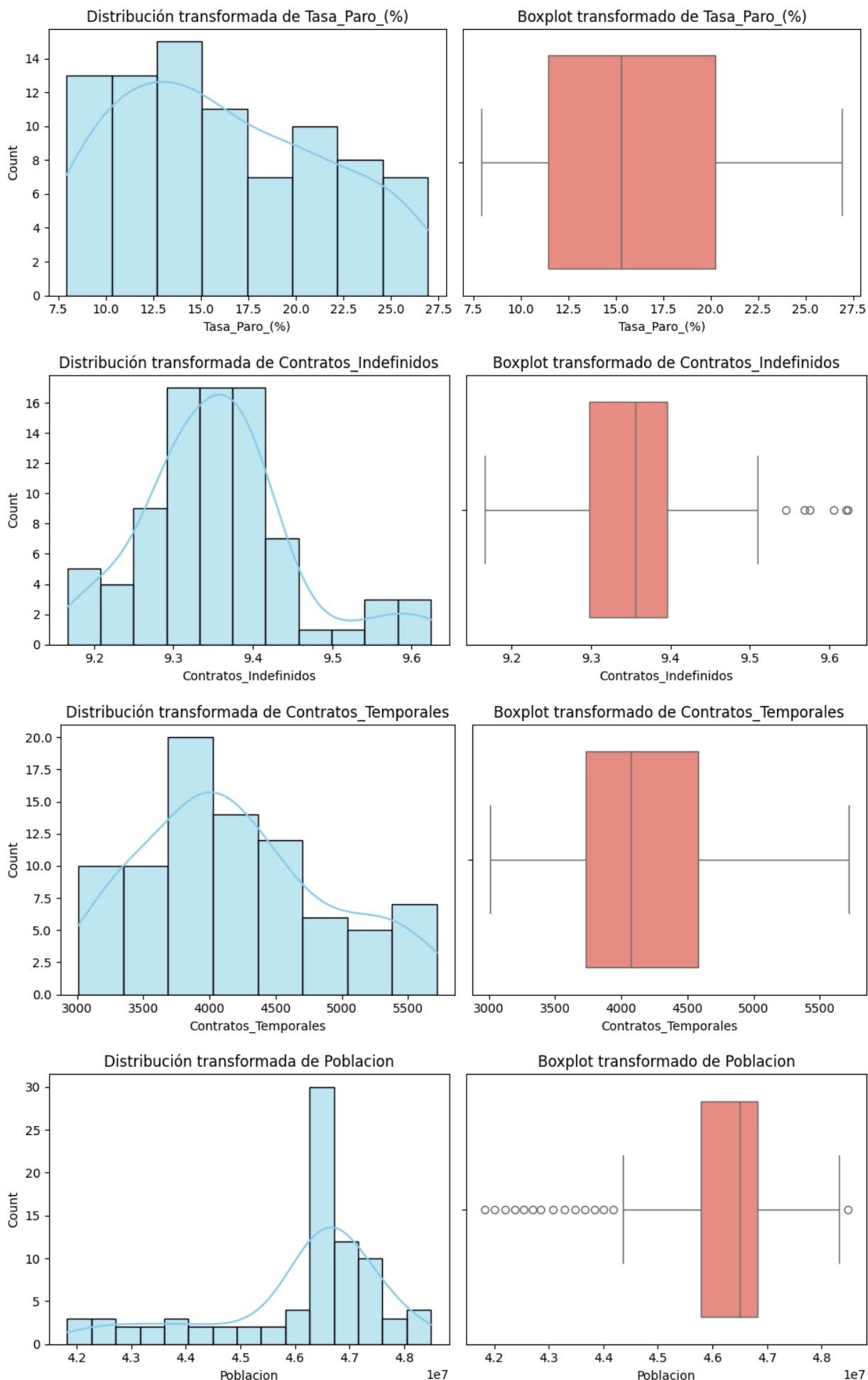
for col in num_vars:
    plt.figure(figsize=(10, 4))

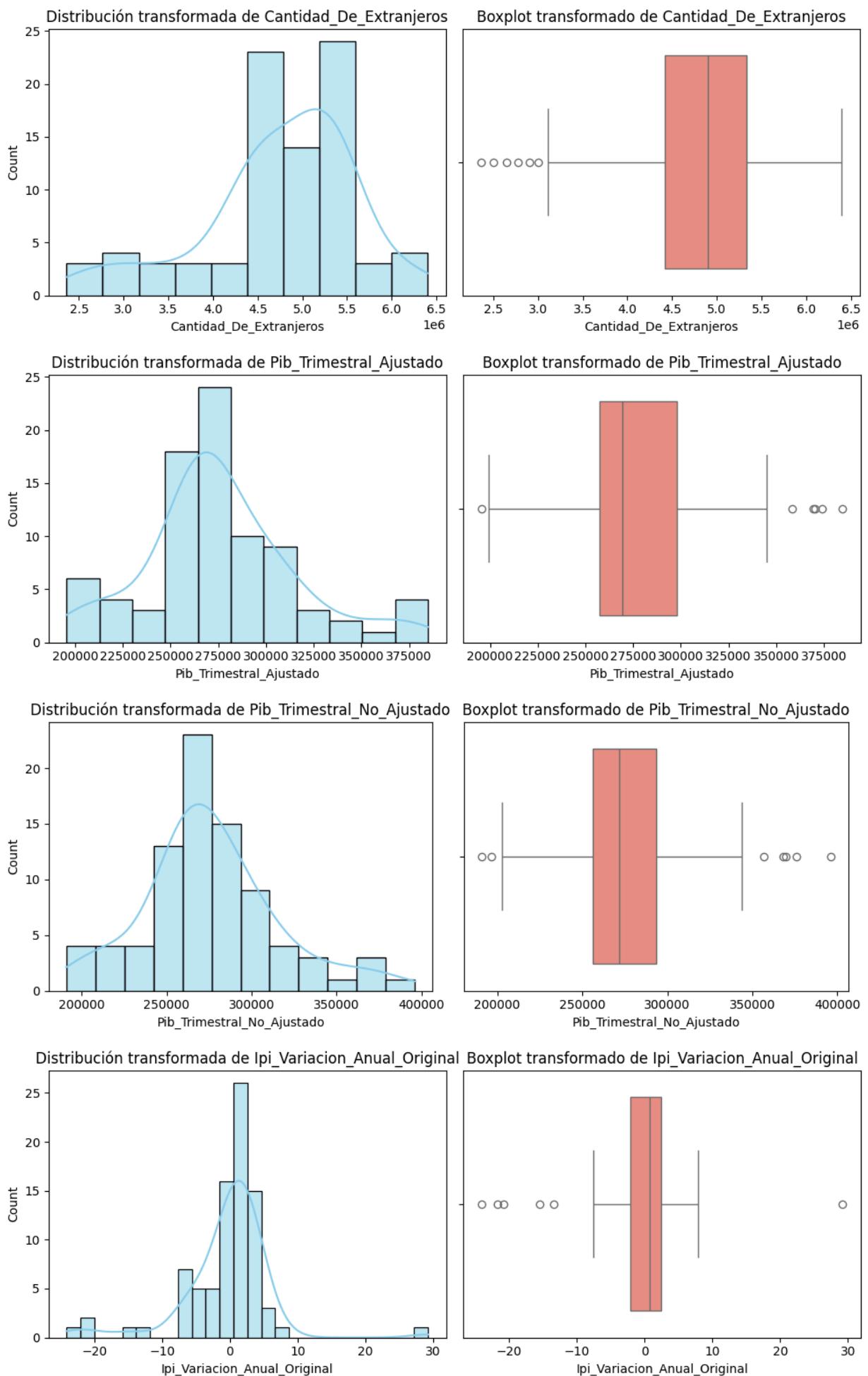
    plt.subplot(1, 2, 1)
    sns.histplot(df_transformed[col], kde=True, color='skyblue')
    plt.title(f'Distribución transformada de {col}')

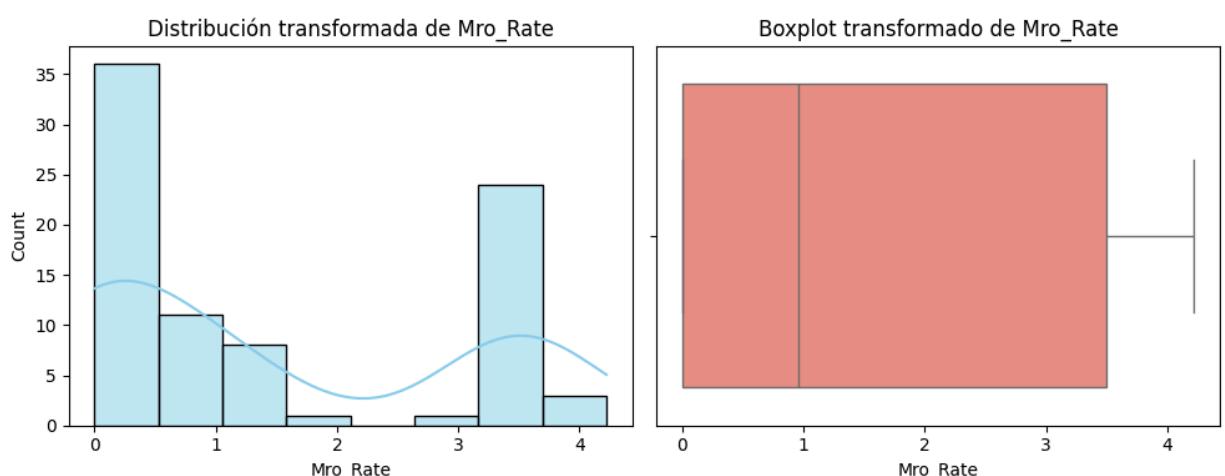
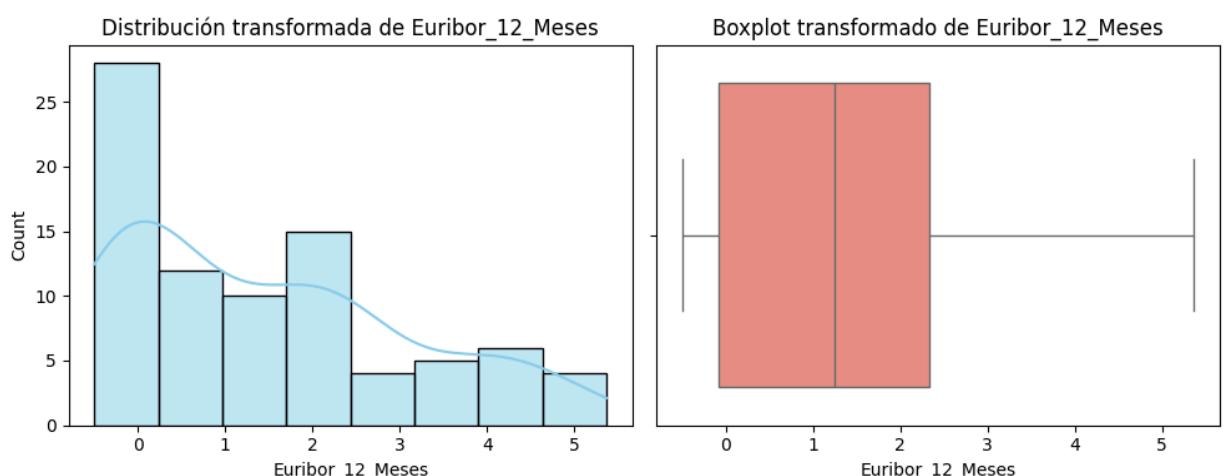
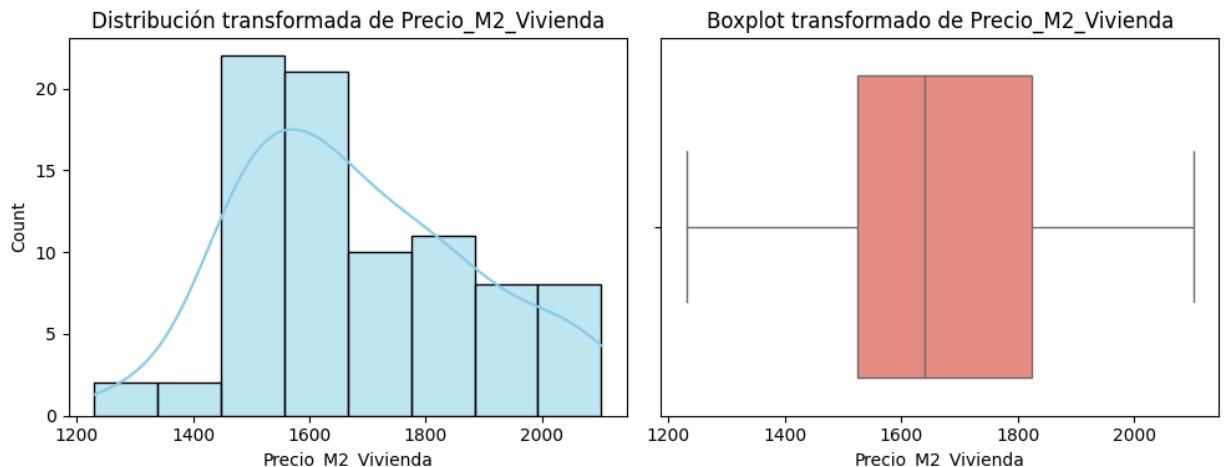
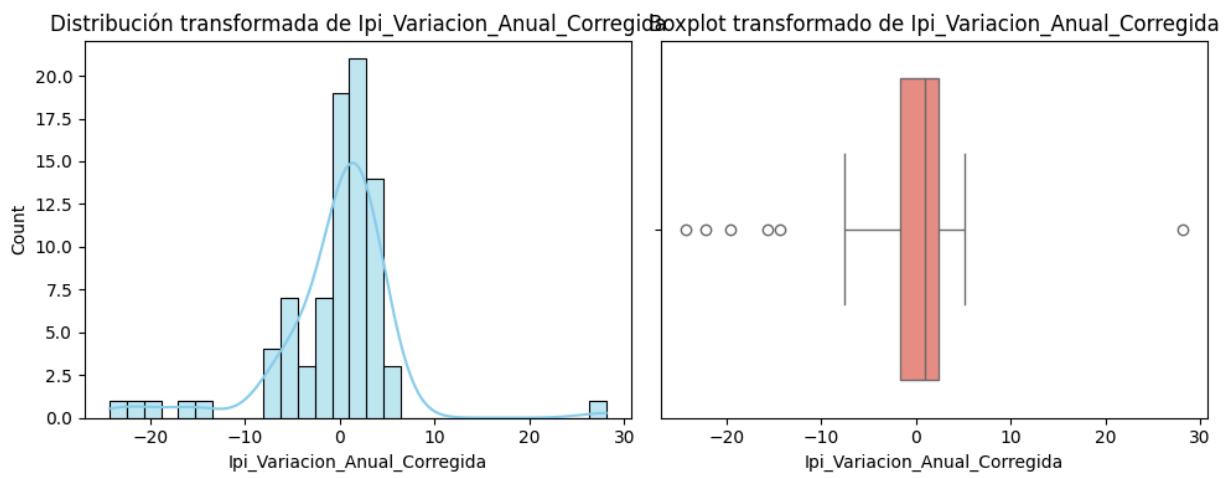
    plt.subplot(1, 2, 2)
    sns.boxplot(x=df_transformed[col], color='salmon')
    plt.title(f'Boxplot transformado de {col}')

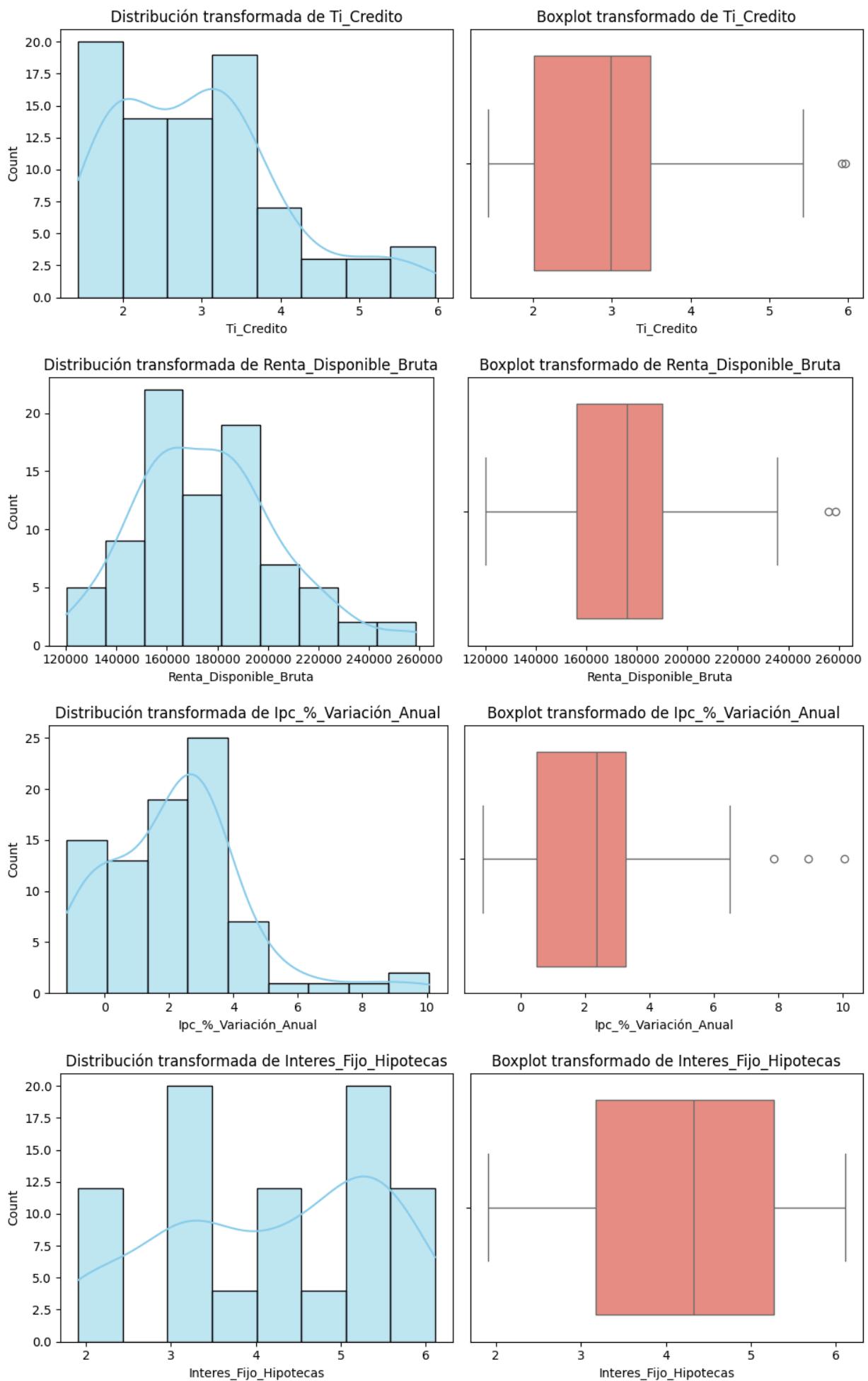
    plt.tight_layout()
    plt.show()
```

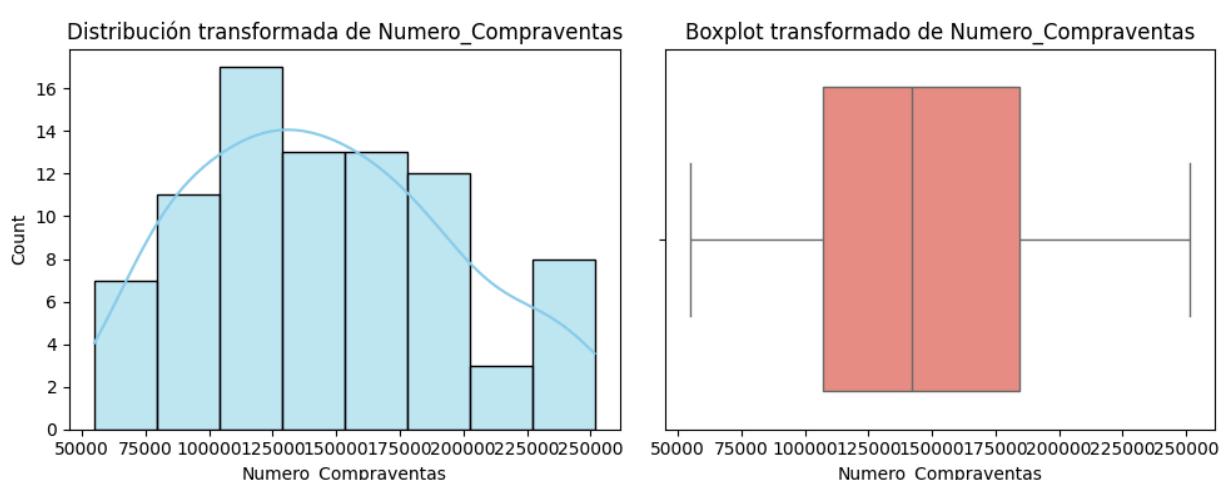
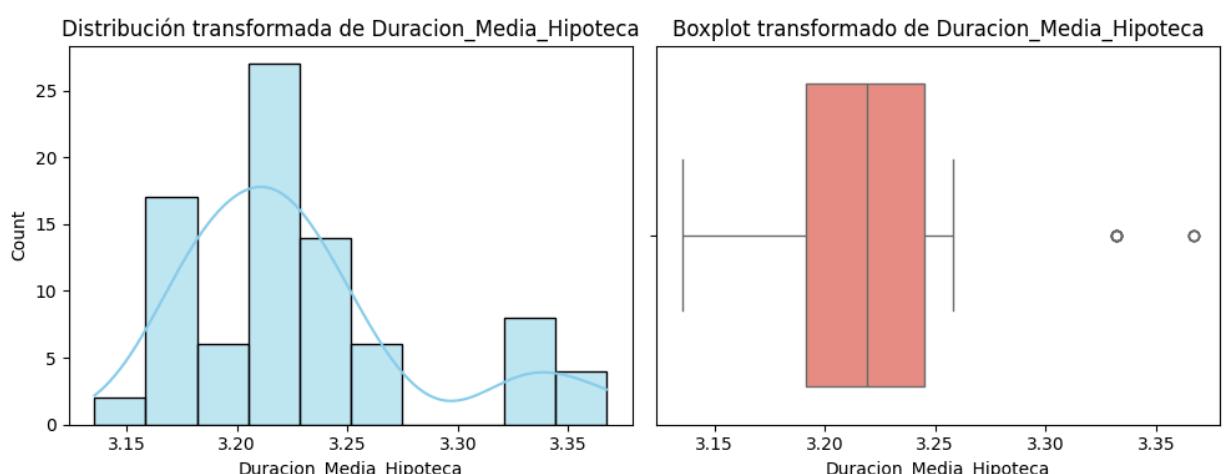
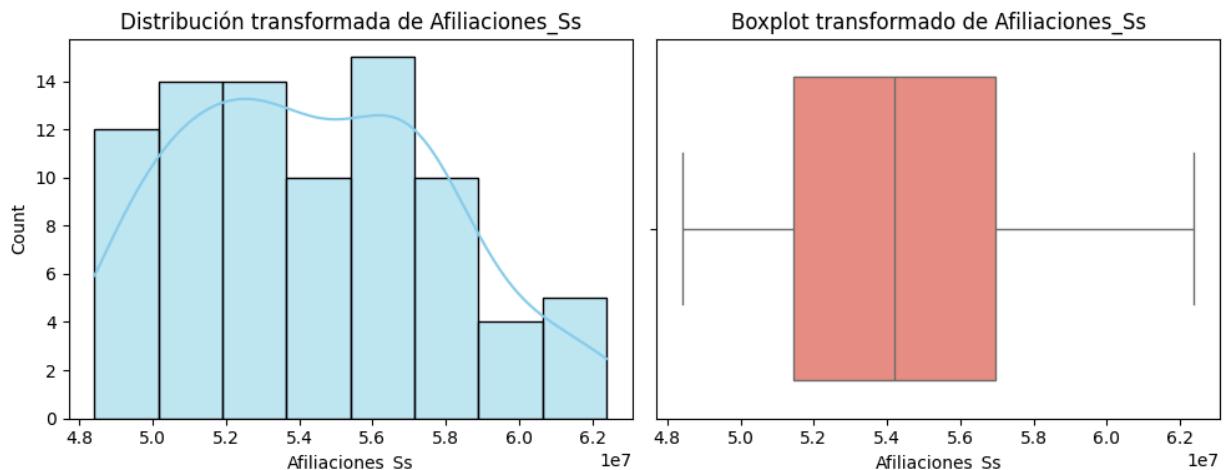
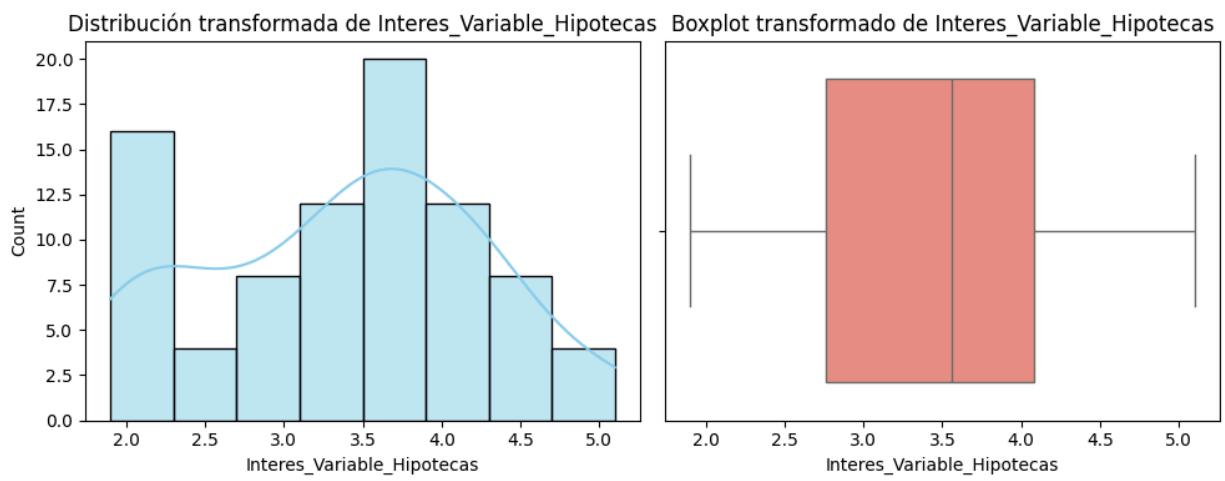


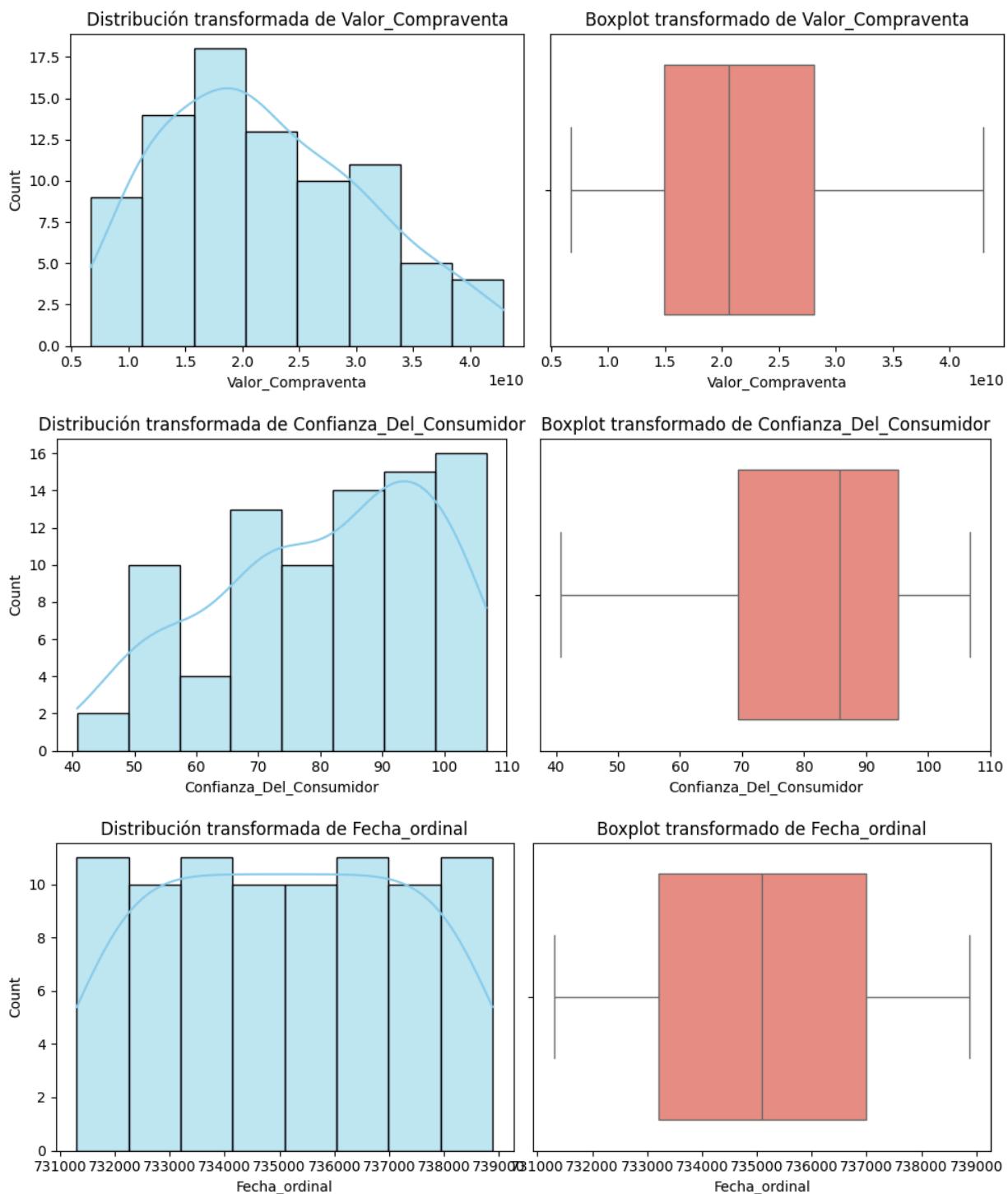












```
In [28]: print("\n◆ Outliers detectados con IQR en df_transformed:")
outliers_iqr_transformed = {}

for col in num_vars:
    Q1 = df_transformed[col].quantile(0.25)
    Q3 = df_transformed[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df_transformed[(df_transformed[col] < lower_bound) | (df_transformed[col] > upper_bound)]
    if not outliers.empty:
        outliers_iqr_transformed[col] = outliers.shape[0]
        print(f'{col}: {outliers.shape[0]} valores atípicos')
```

```
❖ Outliers detectados con IQR en df_transformed:
Contratos_Definidos: 6 valores atípicos
Poblacion: 15 valores atípicos
Cantidad_De_Extranjeros: 6 valores atípicos
Pib_Trimestral_Ajustado: 6 valores atípicos
Pib_Trimestral_No_Ajustado: 7 valores atípicos
Ipi_Variacion_Anual_Original: 6 valores atípicos
Ipi_Variacion_Anual_Corregida: 6 valores atípicos
Ti_Credito: 2 valores atípicos
Renta_Disponible_Bruta: 2 valores atípicos
Ipc_%_Variación_Anual: 3 valores atípicos
Duracion_Media_Hipoteca: 12 valores atípicos
```

Vemos que con estas transformaciones hemos reducido una variable que anteriormente presentaba outliers (antes eran 12 y ahora 11). Esta variable es Importe_Hipotecas

Tratamiento de outliers

Es necesario realizar un tratamiento de los outliers de nuestras variables para los posteriores modelos de regresión lineal múltiples o de series temporales, ya que un solo valor atípico puede tener un gran efecto en la estimación de los coeficientes o distorsionar los modelos. Para ello, vamos a utilizar la técnica de Winsorización (recorte de valores extremos), usando los límites del IQR para asegurarnos de que ningún valor queda por encima o por debajo del rango intercuartílico.

```
In [29]: iqr_winsorized_df = df_transformed.copy()

vars_num = df_transformed.select_dtypes(include='number').columns.drop(['Número_Hipotecas', 'Fecha_ordinal', 'Año'])

for col in vars_num:
    col_data = df_transformed[col].dropna()
    Q1 = col_data.quantile(0.25)
    Q3 = col_data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    iqr_winsorized_df[col] = df_transformed[col].clip(lower=lower_bound, upper=upper_bound)

print("✅ Winsorización corregida y aplicada sin warnings.")
```

✅ Winsorización corregida y aplicada sin warnings.

Comprobamos, una vez aplicada la técnica de Winsorización, cuántos outliers tenemos ahora:

```
In [30]: print("\n❖ Outliers con IQR tras winsorización:")
outliers_iqr_winsor = {}

for col in num_vars:
    Q1 = iqr_winsorized_df[col].quantile(0.25)
    Q3 = iqr_winsorized_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = iqr_winsorized_df[(iqr_winsorized_df[col] < lower_bound) | (iqr_winsorized_df[col] > upper_bound)]

    if not outliers.empty:
        outliers_iqr_winsor[col] = outliers.shape[0]
        print(f"{col}: {outliers.shape[0]} valores atípicos")
    else:
        print(f"{col}: 0 valores atípicos")
```

❖ Outliers con IQR tras winsorización:

Numero_Hipotecas: 0 valores atípicos
Importe_Hipotecas: 0 valores atípicos
Valor_Medio_Hipotecas: 0 valores atípicos
Tipo_Interes_Medio: 0 valores atípicos
Tasa_Paro_(%): 0 valores atípicos
Contratos_Indefinidos: 0 valores atípicos
Contratos_Temporales: 0 valores atípicos
Poblacion: 0 valores atípicos
Cantidad_De_Extranjeros: 0 valores atípicos
Pib_Trimestral_Ajustado: 0 valores atípicos
Pib_Trimestral_No_Ajustado: 0 valores atípicos
Ipi_Variacion_Anual_Original: 0 valores atípicos
Ipi_Variacion_Anual_Corregida: 0 valores atípicos
Precio_M2_Vivienda: 0 valores atípicos
Euribor_12_Meses: 0 valores atípicos
Mro_Rate: 0 valores atípicos
Ti_Credito: 0 valores atípicos
Renta_Disponible_Bruta: 0 valores atípicos
Ipc_%_Variación_Anual: 0 valores atípicos
Interes_Fijo_Hipotecas: 0 valores atípicos
Interes_Variable_Hipotecas: 0 valores atípicos
Afiliaciones_Ss: 0 valores atípicos
Duracion_Media_Hipoteca: 0 valores atípicos
Numero_Compraventas: 0 valores atípicos
Valor_Compraventa: 0 valores atípicos
Confianza_Del_Consumidor: 0 valores atípicos
Fecha_ordinal: 0 valores atípicos

Como vemos, ya ninguna variable presenta outliers

5. Correlaciones y análisis bivariante

Correlación entre variables explicativas

El objetivo es detectar colinealidad (dos o más variables muy correlacionadas entre sí) que pueda perjudicar los posteriores modelos. Vamos a analizarlo a través de una matriz de correlaciones.

Si dos variables tienen correlación > 0.7 o < -0.7 , puede haber problema de multicolinealidad. En este caso, conviene eliminar una, combinarlas o usar técnicas como PCA.

```
In [31]: vars_exp = iqr_winsorized_df.columns.drop(['Numero_Hipotecas', 'Fecha', 'Fecha_ordinal', 'Trimestre_simple', 'Año'])

corr_pearson = iqr_winsorized_df[vars_exp].corr(method='pearson')

corr_spearman = iqr_winsorized_df[vars_exp].corr(method='spearman')

print("Matriz de correlación Pearson (variables explicativas):")
print(corr_pearson)

print("\nMatriz de correlación Spearman (variables explicativas):")
print(corr_spearman)
```

Matriz de correlación Pearson (variables explicativas):

	Importe_Hipotecas	Valor_Medio_Hipotecas	Contratos_Indefinidos	Contratos_Temporales
Importe_Hipotecas	1.000000	0.554021	-0.112153	0.816110
Valor_Medio_Hipotecas	0.554021	1.000000	0.662962	0.391151
Tipo_Interes_Medio	0.225704	-0.272000	-0.463076	0.199724
Tasa_Paro_(%)	-0.873006	-0.689419	-0.085794	-0.768760
Contratos_Indefinidos	-0.112153	0.662962	1.000000	-0.355168
Contratos_Temporales	0.816110	0.391151	0.199724	1.000000
Poblacion	-0.632689	0.177489	-0.355168	-0.755124
Cantidad_De_Extranjeros	-0.300514	0.427521	0.886380	0.244893
Pib_Trimestral_Ajustado	-0.252105	0.576205	0.244893	0.661824
Pib_Trimestral_No_Ajustado	-0.266238	0.556155	0.244893	0.741299
Ipi_Variacion_Anual_Original	0.117703	0.071244	-0.158777	0.227147
Ipi_Variacion_Anual_Corregida	0.132435	0.088638	0.244893	0.672471
Precio_M2_Vivienda	0.628499	0.737781	0.244893	0.160040
Euribor_12_Meses	0.685924	0.385595	0.244893	0.536398
Mro_Rate	0.774057	0.277147	0.244893	0.382027
Ti_Credito	0.522893	0.187656	0.244893	0.271045
Renta_Disponible_Bruta	-0.285107	0.444324	0.244893	0.053653
Ipc_%_Variación_Anual	0.500424	0.450583	0.244893	0.057111
Interes_Fijo_Hipotecas	0.067218	-0.427596	0.244893	0.244893
Interes_Variable_Hipotecas	0.267563	-0.272780	0.244893	0.244893
Afiliaciones_Ss	0.351628	0.910211	0.244893	0.244893
Duracion_Media_Hipoteca	0.775686	0.757787	0.244893	0.244893
Numero_Compraventas	0.804782	0.516456	0.244893	0.244893
Valor_Compraventa	0.795879	0.754640	0.244893	0.244893
Confianza_Del_Consumidor	0.130926	-0.144231	0.244893	0.244893
	Tipo_Interes_Medio	Tasa_Paro_(%)		
Importe_Hipotecas	0.225704	-0.873006		
Valor_Medio_Hipotecas	-0.272000	-0.689419		
Tipo_Interes_Medio	1.000000	0.053653		
Tasa_Paro_(%)	0.053653	1.000000		
Contratos_Indefinidos	-0.463076	-0.085794		
Contratos_Temporales	0.199724	-0.768760		
Poblacion	-0.496335	0.443543		
Cantidad_De_Extranjeros	-0.221019	0.256024		
Pib_Trimestral_Ajustado	-0.547220	-0.015200		
Pib_Trimestral_No_Ajustado	-0.544185	-0.010186		
Ipi_Variacion_Anual_Original	-0.323683	-0.210294		
Ipi_Variacion_Anual_Corregida	-0.354301	-0.231266		
Precio_M2_Vivienda	0.284848	-0.477981		
Euribor_12_Meses	0.655939	-0.552809		
Mro_Rate	0.539199	-0.678275		
Ti_Credito	0.852046	-0.346050		
Renta_Disponible_Bruta	-0.511737	0.027683		
Ipc_%_Variación_Anual	-0.026005	-0.492679		
Interes_Fijo_Hipotecas	0.931611	0.269157		
Interes_Variable_Hipotecas	0.973171	0.043739		
Afiliaciones_Ss	-0.429744	-0.597704		
Duracion_Media_Hipoteca	0.247226	-0.679093		
Numero_Compraventas	-0.077986	-0.866513		
Valor_Compraventa	-0.137090	-0.855247		
Confianza_Del_Consumidor	-0.190985	-0.283954		
	Contratos_Indefinidos	Contratos_Temporales		
Importe_Hipotecas	-0.112153	0.816110		
Valor_Medio_Hipotecas	0.662962	0.391151		
Tipo_Interes_Medio	-0.463076	0.199724		
Tasa_Paro_(%)	-0.085794	-0.768760		
Contratos_Indefinidos	1.000000	-0.355168		
Contratos_Temporales	-0.355168	1.000000		
Poblacion	0.822556	-0.757351		
Cantidad_De_Extranjeros	0.887755	-0.547375		
Pib_Trimestral_Ajustado	0.948528	-0.383207		
Pib_Trimestral_No_Ajustado	0.930093	-0.363322		
Ipi_Variacion_Anual_Original	-0.158777	0.260562		
Ipi_Variacion_Anual_Corregida	-0.151765	0.270736		
Precio_M2_Vivienda	0.420450	0.425594		
Euribor_12_Meses	-0.022826	0.427495		
Mro_Rate	-0.236337	0.536398		
Ti_Credito	-0.187537	0.388471		
Renta_Disponible_Bruta	0.832336	-0.398613		
Ipc_%_Variación_Anual	0.214078	0.271045		
Interes_Fijo_Hipotecas	-0.497591	0.057111		
Interes_Variable_Hipotecas	-0.505766	0.244893		
Afiliaciones_Ss	0.836421	0.160040		
Duracion_Media_Hipoteca	0.186314	0.661824		
Numero_Compraventas	-0.107124	0.741299		
Valor_Compraventa	0.185054	0.672471		

Confianza_Del_Consumidor	-0.368072	0.363972
	Poblacion	Cantidad_De_Extranjeros \
Importe_Hipotecas	-0.632689	-0.300514
Valor_Medio_Hipotecas	0.177489	0.427521
Tipo_Interes_Medio	-0.496335	-0.221019
Tasa_Paro_(%)	0.443543	0.256024
Contratos_Indefinidos	0.822556	0.887755
Contratos_Temporales	-0.757351	-0.547375
Poblacion	1.000000	0.869983
Cantidad_De_Extranjeros	0.869983	1.000000
Pib_Trimestral_Ajustado	0.838247	0.798273
Pib_Trimestral_No_Ajustado	0.825017	0.781394
Ipi_Variacion_Anual_Original	-0.217615	-0.370426
Ipi_Variacion_Anual_Corregida	-0.223773	-0.379966
Precio_M2_Vivienda	-0.041049	0.420006
Euribor_12_Meses	-0.420524	-0.035342
Mro_Rate	-0.618724	-0.317487
Ti_Credito	-0.459943	-0.091991
Renta_Disponible_Bruta	0.774151	0.727979
Ipc_%_Variación_Anual	-0.103945	0.064871
Interes_Fijo_Hipotecas	-0.422984	-0.184214
Interes_Variable_Hipotecas	-0.545463	-0.258682
Afilaciones_Ss	0.424002	0.563732
Duracion_Media_Hipoteca	-0.306206	0.087904
Numero_Compraventas	-0.558823	-0.409103
Valor_Compraventa	-0.331190	-0.091977
Confianza_Del_Consumidor	-0.397656	-0.627201
	Pib_Trimestral_Ajustado \	
Importe_Hipotecas	-0.252105	
Valor_Medio_Hipotecas	0.576205	
Tipo_Interes_Medio	-0.547220	
Tasa_Paro_(%)	-0.015200	
Contratos_Indefinidos	0.948528	
Contratos_Temporales	-0.383207	
Poblacion	0.838247	
Cantidad_De_Extranjeros	0.798273	
Pib_Trimestral_Ajustado	1.000000	
Pib_Trimestral_No_Ajustado	0.979015	
Ipi_Variacion_Anual_Original	0.002806	
Ipi_Variacion_Anual_Corregida	0.010268	
Precio_M2_Vivienda	0.246119	
Euribor_12_Meses	-0.160972	
Mro_Rate	-0.345938	
Ti_Credito	-0.305563	
Renta_Disponible_Bruta	0.835896	
Ipc_%_Variación_Anual	0.166974	
Interes_Fijo_Hipotecas	-0.576298	
Interes_Variable_Hipotecas	-0.600608	
Afilaciones_Ss	0.781814	
Duracion_Media_Hipoteca	0.059305	
Numero_Compraventas	-0.124598	
Valor_Compraventa	0.133497	
Confianza_Del_Consumidor	-0.183345	
	Pib_Trimestral_No_Ajustado ... Ti_Credito \	
Importe_Hipotecas	-0.266238 ... 0.522893	
Valor_Medio_Hipotecas	0.556155 ... 0.187656	
Tipo_Interes_Medio	-0.544185 ... 0.852046	
Tasa_Paro_(%)	-0.010186 ... -0.346050	
Contratos_Indefinidos	0.930093 ... -0.187537	
Contratos_Temporales	-0.363322 ... 0.388471	
Poblacion	0.825017 ... -0.459943	
Cantidad_De_Extranjeros	0.781394 ... -0.091991	
Pib_Trimestral_Ajustado	0.979015 ... -0.305563	
Pib_Trimestral_No_Ajustado	1.000000 ... -0.318307	
Ipi_Variacion_Anual_Original	0.007550 ... -0.259388	
Ipi_Variacion_Anual_Corregida	0.014028 ... -0.281338	
Precio_M2_Vivienda	0.237246 ... 0.570073	
Euribor_12_Meses	-0.179241 ... 0.926461	
Mro_Rate	-0.363894 ... 0.798552	
Ti_Credito	-0.318307 ... 1.000000	
Renta_Disponible_Bruta	0.908029 ... -0.322607	
Ipc_%_Variación_Anual	0.157397 ... 0.284843	
Interes_Fijo_Hipotecas	-0.565742 ... 0.694620	
Interes_Variable_Hipotecas	-0.591038 ... 0.822872	
Afilaciones_Ss	0.769728 ... 0.009009	
Duracion_Media_Hipoteca	0.050688 ... 0.597778	
Numero_Compraventas	-0.086740 ... 0.238142	

Valor_Compraventa	0.163087	...	0.256441
Confianza_Del_Consumidor	-0.181626	...	-0.212926
Renta_Disponible_Bruta Ipc_%_Variación_Anual \			
Importe_Hipotecas	-0.285107	0.500424	
Valor_Medio_Hipotecas	0.444324	0.450583	
Tipo_Interes_Medio	-0.511737	-0.026005	
Tasa_Paro_(%)	0.027683	-0.492679	
Contratos_Indefinidos	0.832336	0.214078	
Contratos_Temporales	-0.398613	0.271045	
Poblacion	0.774151	-0.103945	
Cantidad_De_Extranjeros	0.727979	0.064871	
Pib_Trimestral_Ajustado	0.835896	0.166974	
Pib_Trimestral_No_Ajustado	0.908029	0.157397	
Ipi_Variacion_Anual_Original	-0.065165	0.150915	
Ipi_Variacion_Anual_Corregida	-0.050833	0.131667	
Precio_M2_Vivienda	0.177090	0.348575	
Euribor_12_Meses	-0.186070	0.460112	
Mro_Rate	-0.335219	0.459955	
Ti_Credito	-0.322607	0.284843	
Renta_Disponible_Bruta	1.000000	0.081771	
Ipc_%_Variación_Anual	0.081771	1.000000	
Interes_Fijo_Hipotecas	-0.518129	-0.140099	
Interes_Variable_Hipotecas	-0.550455	-0.013966	
Afiliaciones_Ss	0.673898	0.413263	
Duracion_Media_Hipoteca	-0.011983	0.435141	
Numero_Compraventas	-0.065918	0.510043	
Valor_Compraventa	0.148758	0.556789	
Confianza_Del_Consumidor	-0.222616	-0.140845	
Interes_Fijo_Hipotecas \			
Importe_Hipotecas	0.067218		
Valor_Medio_Hipotecas	-0.427596		
Tipo_Interes_Medio	0.931611		
Tasa_Paro_(%)	0.269157		
Contratos_Indefinidos	-0.497591		
Contratos_Temporales	0.057111		
Poblacion	-0.422984		
Cantidad_De_Extranjeros	-0.184214		
Pib_Trimestral_Ajustado	-0.576298		
Pib_Trimestral_No_Ajustado	-0.565742		
Ipi_Variacion_Anual_Original	-0.312293		
Ipi_Variacion_Anual_Corregida	-0.342836		
Precio_M2_Vivienda	0.186933		
Euribor_12_Meses	0.482870		
Mro_Rate	0.360957		
Ti_Credito	0.694620		
Renta_Disponible_Bruta	-0.518129		
Ipc_%_Variación_Anual	-0.140099		
Interes_Fijo_Hipotecas	1.000000		
Interes_Variable_Hipotecas	0.950580		
Afiliaciones_Ss	-0.575785		
Duracion_Media_Hipoteca	0.082413		
Numero_Compraventas	-0.241090		
Valor_Compraventa	-0.297158		
Confianza_Del_Consumidor	-0.209830		
Interes_Variable_Hipotecas Afiliaciones_Ss \			
Importe_Hipotecas	0.267563	0.351628	
Valor_Medio_Hipotecas	-0.272780	0.910211	
Tipo_Interes_Medio	0.973171	-0.429744	
Tasa_Paro_(%)	0.043739	-0.597704	
Contratos_Indefinidos	-0.505766	0.836421	
Contratos_Temporales	0.244893	0.160040	
Poblacion	-0.545463	0.424002	
Cantidad_De_Extranjeros	-0.258682	0.563732	
Pib_Trimestral_Ajustado	-0.600608	0.781814	
Pib_Trimestral_No_Ajustado	-0.591038	0.769728	
Ipi_Variacion_Anual_Original	-0.294348	0.025139	
Ipi_Variacion_Anual_Corregida	-0.324942	0.041671	
Precio_M2_Vivienda	0.294370	0.579850	
Euribor_12_Meses	0.642096	0.238406	
Mro_Rate	0.542810	0.139247	
Ti_Credito	0.822872	0.009009	
Renta_Disponible_Bruta	-0.550455	0.673898	
Ipc_%_Variación_Anual	-0.013966	0.413263	
Interes_Fijo_Hipotecas	0.950580	-0.575785	
Interes_Variable_Hipotecas	1.000000	-0.459030	
Afiliaciones_Ss	-0.459030	1.000000	
Duracion_Media_Hipoteca	0.267978	0.522532	

Numero_Compraventas	-0.048261	0.390329
Valor_Compraventa	-0.111073	0.622325
Confianza_Del_Consumidor	-0.175094	-0.106176

	Duracion_Media_Hipoteca	Numero_Compraventas \
Importe_Hipotecas	0.775686	0.804782
Valor_Medio_Hipotecas	0.757787	0.516456
Tipo_Interes_Medio	0.247226	-0.077986
Tasa_Paro_(%)	-0.679093	-0.866513
Contratos_Indefinidos	0.186314	-0.107124
Contratos_Temporales	0.661824	0.741299
Poblacion	-0.306206	-0.558823
Cantidad_De_Extranjeros	0.087904	-0.409103
Pib_Trimestral_Ajustado	0.059305	-0.124598
Pib_Trimestral_No_Ajustado	0.050688	-0.086740
Ipi_Variacion_Anual_Original	0.010792	0.405194
Ipi_Variacion_Anual_Corregida	0.017280	0.422928
Precio_M2_Vivienda	0.841820	0.322865
Euribor_12_Meses	0.688514	0.436547
Mro_Rate	0.579632	0.632688
Ti_Credito	0.597778	0.238142
Renta_Disponible_Bruta	-0.011983	-0.065918
Ipc_%_Variación_Anual	0.435141	0.510043
Interes_Fijo_Hipotecas	0.082413	-0.241090
Interes_Variable_Hipotecas	0.267978	-0.048261
Afiliaciones_Ss	0.522532	0.390329
Duracion_Media_Hipoteca	1.000000	0.562444
Numero_Compraventas	0.562444	1.000000
Valor_Compraventa	0.714888	0.934071
Confianza_Del_Consumidor	-0.171370	0.345591

	Valor_Compraventa	Confianza_Del_Consumidor
Importe_Hipotecas	0.795879	0.130926
Valor_Medio_Hipotecas	0.754640	-0.144231
Tipo_Interes_Medio	-0.137090	-0.190985
Tasa_Paro_(%)	-0.855247	-0.283954
Contratos_Indefinidos	0.185054	-0.368072
Contratos_Temporales	0.672471	0.363972
Poblacion	-0.331190	-0.397656
Cantidad_De_Extranjeros	-0.091977	-0.627201
Pib_Trimestral_Ajustado	0.133497	-0.183345
Pib_Trimestral_No_Ajustado	0.163087	-0.181626
Ipi_Variacion_Anual_Original	0.332779	0.616777
Ipi_Variacion_Anual_Corregida	0.353288	0.671595
Precio_M2_Vivienda	0.566374	-0.373141
Euribor_12_Meses	0.478131	-0.156477
Mro_Rate	0.564450	0.054555
Ti_Credito	0.256441	-0.212926
Renta_Disponible_Bruta	0.148758	-0.222616
Ipc_%_Variación_Anual	0.556789	-0.140845
Interes_Fijo_Hipotecas	-0.297158	-0.209830
Interes_Variable_Hipotecas	-0.111073	-0.175094
Afiliaciones_Ss	0.622325	-0.106176
Duracion_Media_Hipoteca	0.714888	-0.171370
Numero_Compraventas	0.934071	0.345591
Valor_Compraventa	1.000000	0.158712
Confianza_Del_Consumidor	0.158712	1.000000

[25 rows x 25 columns]

Matriz de correlación Spearman (variables explicativas):

	Importe_Hipotecas	Valor_Medio_Hipotecas \
Importe_Hipotecas	1.000000	0.563552
Valor_Medio_Hipotecas	0.563552	1.000000
Tipo_Interes_Medio	0.203505	-0.320921
Tasa_Paro_(%)	-0.872046	-0.674010
Contratos_Indefinidos	-0.021286	0.660307
Contratos_Temporales	0.750248	0.374203
Poblacion	-0.494072	0.226969
Cantidad_De_Extranjeros	-0.123035	0.398005
Pib_Trimestral_Ajustado	-0.199372	0.553746
Pib_Trimestral_No_Ajustado	-0.233466	0.517368
Ipi_Variacion_Anual_Original	0.114235	0.065973
Ipi_Variacion_Anual_Corregida	0.127456	0.088603
Precio_M2_Vivienda	0.654895	0.761385
Euribor_12_Meses	0.620688	0.190514
Mro_Rate	0.654340	0.122884
Ti_Credito	0.468090	0.028642
Renta_Disponible_Bruta	-0.273242	0.407180
Ipc_%_Variación_Anual	0.583176	0.386938

Interes_Fijo_Hipotecas	0.047109	-0.399575
Interes_Variable_Hipotecas	0.253465	-0.294141
Afiliaciones_Ss	0.405184	0.929655
Duracion_Media_Hipoteca	0.746574	0.793670
Numero_Compraventas	0.768229	0.535810
Valor_Compraventa	0.782869	0.761831
Confianza_Del_Consumidor	0.062144	-0.169626

	Tipo_Interes_Medio	Tasa_Paro_(%)	\
Importe_Hipotecas	0.203505	-0.872046	
Valor_Medio_Hipotecas	-0.320921	-0.674010	
Tipo_Interes_Medio	1.000000	0.021717	
Tasa_Paro_(%)	0.021717	1.000000	
Contratos_Indefinidos	-0.500772	-0.061052	
Contratos_Temporales	0.125313	-0.768148	
Poblacion	-0.593371	0.364808	
Cantidad_De_Extranjeros	-0.216807	0.221250	
Pib_Trimestral_Ajustado	-0.627973	0.009984	
Pib_Trimestral_No_Ajustado	-0.602146	0.029506	
Ipi_Variacion_Anual_Original	-0.250446	-0.184531	
Ipi_Variacion_Anual_Corregida	-0.270018	-0.213345	
Precio_M2_Vivienda	0.159453	-0.497135	
Euribor_12_Meses	0.683406	-0.505369	
Mro_Rate	0.630224	-0.528685	
Ti_Credito	0.820434	-0.357641	
Renta_Disponible_Bruta	-0.542977	0.112616	
Ipc_%_Variación_Anual	0.097902	-0.573759	
Interes_Fijo_Hipotecas	0.938337	0.211301	
Interes_Variable_Hipotecas	0.974735	-0.019709	
Afiliaciones_Ss	-0.474116	-0.571894	
Duracion_Media_Hipoteca	0.020786	-0.674780	
Numero_Compraventas	-0.120291	-0.880105	
Valor_Compraventa	-0.187478	-0.873950	
Confianza_Del_Consumidor	-0.222398	-0.232743	

	Contratos_Indefinidos	Contratos_Temporales	\
Importe_Hipotecas	-0.021286	0.750248	
Valor_Medio_Hipotecas	0.660307	0.374203	
Tipo_Interes_Medio	-0.500772	0.125313	
Tasa_Paro_(%)	-0.061052	-0.768148	
Contratos_Indefinidos	1.000000	-0.218729	
Contratos_Temporales	-0.218729	1.000000	
Poblacion	0.760073	-0.621561	
Cantidad_De_Extranjeros	0.836947	-0.508959	
Pib_Trimestral_Ajustado	0.889212	-0.225557	
Pib_Trimestral_No_Ajustado	0.860444	-0.239086	
Ipi_Variacion_Anual_Original	-0.212715	0.233563	
Ipi_Variacion_Anual_Corregida	-0.231542	0.266507	
Precio_M2_Vivienda	0.472484	0.337552	
Euribor_12_Meses	-0.254962	0.271238	
Mro_Rate	-0.334075	0.284782	
Ti_Credito	-0.341856	0.238758	
Renta_Disponible_Bruta	0.780339	-0.341249	
Ipc_%_Variación_Anual	0.018726	0.345513	
Interes_Fijo_Hipotecas	-0.447249	-0.032190	
Interes_Variable_Hipotecas	-0.507466	0.186875	
Afiliaciones_Ss	0.822764	0.253944	
Duracion_Media_Hipoteca	0.324100	0.494347	
Numero_Compraventas	-0.045953	0.687213	
Valor_Compraventa	0.225139	0.614701	
Confianza_Del_Consumidor	-0.374532	0.389491	

	Poblacion	Cantidad_De_Extranjeros	\
Importe_Hipotecas	-0.494072	-0.123035	
Valor_Medio_Hipotecas	0.226969	0.398005	
Tipo_Interes_Medio	-0.593371	-0.216807	
Tasa_Paro_(%)	0.364808	0.221250	
Contratos_Indefinidos	0.760073	0.836947	
Contratos_Temporales	-0.621561	-0.508959	
Poblacion	1.000000	0.764934	
Cantidad_De_Extranjeros	0.764934	1.000000	
Pib_Trimestral_Ajustado	0.728026	0.614842	
Pib_Trimestral_No_Ajustado	0.734646	0.608363	
Ipi_Variacion_Anual_Original	-0.207373	-0.395521	
Ipi_Variacion_Anual_Corregida	-0.232204	-0.424917	
Precio_M2_Vivienda	0.006342	0.470743	
Euribor_12_Meses	-0.511245	-0.116028	
Mro_Rate	-0.532261	-0.153591	
Ti_Credito	-0.520429	-0.172351	
Renta_Disponible_Bruta	0.723551	0.635936	

Ipc_%_Variación_Anual	-0.081742	0.033882
Interes_Fijo_Hipotecas	-0.521997	-0.118077
Interes_Variable_Hipotecas	-0.640493	-0.238827
Afiliaciones_Ss	0.431957	0.512858
Duracion_Media_Hipoteca	-0.101752	0.232716
Numero_Compraventas	-0.334111	-0.280499
Valor_Compraventa	-0.150372	0.000182
Confianza_Del_Consumidor	-0.399179	-0.692205

	Pib_Trimestral_Ajustado	\
Importe_Hipotecas	-0.199372	
Valor_Medio_Hipotecas	0.553746	
Tipo_Interes_Medio	-0.627973	
Tasa_Paro_(%)	0.009984	
Contratos_Indefinidos	0.889212	
Contratos_Temporales	-0.225557	
Poblacion	0.728026	
Cantidad_De_Extranjeros	0.614842	
Pib_Trimestral_Ajustado	1.000000	
Pib_Trimestral_No_Ajustado	0.950486	
Ipi_Variacion_Anual_Original	0.010917	
Ipi_Variacion_Anual_Corregida	-0.001418	
Precio_M2_Vivienda	0.229425	
Euribor_12_Meses	-0.395111	
Mro_Rate	-0.480076	
Ti_Credito	-0.467418	
Renta_Disponible_Bruta	0.767400	
Ipc_%_Variación_Anual	-0.045358	
Interes_Fijo_Hipotecas	-0.567473	
Interes_Variable_Hipotecas	-0.641555	
Afiliaciones_Ss	0.747853	
Duracion_Media_Hipoteca	0.145906	
Numero_Compraventas	-0.054192	
Valor_Compraventa	0.154577	
Confianza_Del_Consumidor	-0.073795	

	Pib_Trimestral_No_Ajustado	...	Ti_Credito	\
Importe_Hipotecas	-0.233466	...	0.468090	
Valor_Medio_Hipotecas	0.517368	...	0.028642	
Tipo_Interes_Medio	-0.602146	...	0.820434	
Tasa_Paro_(%)	0.029506	...	-0.357641	
Contratos_Indefinidos	0.860444	...	-0.341856	
Contratos_Temporales	-0.239086	...	0.238758	
Poblacion	0.734646	...	-0.520429	
Cantidad_De_Extranjeros	0.608363	...	-0.172351	
Pib_Trimestral_Ajustado	0.950486	...	-0.467418	
Pib_Trimestral_No_Ajustado	1.000000	...	-0.460809	
Ipi_Variacion_Anual_Original	-0.003342	...	-0.223508	
Ipi_Variacion_Anual_Corregida	-0.006785	...	-0.228140	
Precio_M2_Vivienda	0.212172	...	0.391781	
Euribor_12_Meses	-0.390729	...	0.935557	
Mro_Rate	-0.469652	...	0.884854	
Ti_Credito	-0.460809	...	1.000000	
Renta_Disponible_Bruta	0.889043	...	-0.433466	
Ipc_%_Variación_Anual	-0.039211	...	0.401410	
Interes_Fijo_Hipotecas	-0.541040	...	0.686853	
Interes_Variable_Hipotecas	-0.613126	...	0.799372	
Afiliaciones_Ss	0.722665	...	-0.125879	
Duracion_Media_Hipoteca	0.111709	...	0.316043	
Numero_Compraventas	-0.000516	...	0.191031	
Valor_Compraventa	0.196619	...	0.162773	
Confianza_Del_Consumidor	-0.097641	...	-0.172817	

	Renta_Disponible_Bruta	Ipc_%_Variación_Anual	\
Importe_Hipotecas	-0.273242	0.583176	
Valor_Medio_Hipotecas	0.407180	0.386938	
Tipo_Interes_Medio	-0.542977	0.097902	
Tasa_Paro_(%)	0.112616	-0.573759	
Contratos_Indefinidos	0.780339	0.018726	
Contratos_Temporales	-0.341249	0.345513	
Poblacion	0.723551	-0.081742	
Cantidad_De_Extranjeros	0.635936	0.033882	
Pib_Trimestral_Ajustado	0.767400	-0.045358	
Pib_Trimestral_No_Ajustado	0.889043	-0.039211	
Ipi_Variacion_Anual_Original	-0.097101	0.083243	
Ipi_Variacion_Anual_Corregida	-0.085180	0.058374	
Precio_M2_Vivienda	0.194139	0.416566	
Euribor_12_Meses	-0.359455	0.540174	
Mro_Rate	-0.418238	0.560862	
Ti_Credito	-0.433466	0.401410	

Renta_Disponible_Bruta	1.000000	-0.078192
Ipc_%_Variación_Anual	-0.078192	1.000000
Interes_Fijo_Hipotecas	-0.469389	-0.065077
Interes_Variable_Hipotecas	-0.554828	0.085500
Afiliaciones_Ss	0.606716	0.316017
Duracion_Media_Hipoteca	0.071447	0.520953
Numero_Compraventas	-0.038089	0.574073
Valor_Compraventa	0.154278	0.577070
Confianza_Del_Consumidor	-0.225647	-0.115981

	Interes_Fijo_Hipotecas	\
Importe_Hipotecas	0.047109	
Valor_Medio_Hipotecas	-0.399575	
Tipo_Interes_Medio	0.938337	
Tasa_Paro_(%)	0.211301	
Contratos_Indefinidos	-0.447249	
Contratos_Temporales	-0.032190	
Poblacion	-0.521997	
Cantidad_De_Extranjeros	-0.118077	
Pib_Trimestral_Ajustado	-0.567473	
Pib_Trimestral_No_Ajustado	-0.541040	
Ipi_Variacion_Anual_Original	-0.287527	
Ipi_Variacion_Anual_Corregida	-0.312335	
Precio_M2_Vivienda	0.111488	
Euribor_12_Meses	0.557850	
Mro_Rate	0.497999	
Ti_Credito	0.686853	
Renta_Disponible_Bruta	-0.469389	
Ipc_%_Variación_Anual	-0.065077	
Interes_Fijo_Hipotecas	1.000000	
Interes_Variable_Hipotecas	0.951608	
Afiliaciones_Ss	-0.531009	
Duracion_Media_Hipoteca	-0.085913	
Numero_Compraventas	-0.304384	
Valor_Compraventa	-0.333898	
Confianza_Del_Consumidor	-0.273653	

	Interes_Variable_Hipotecas	Afiliaciones_Ss	\
Importe_Hipotecas	0.253465	0.405184	
Valor_Medio_Hipotecas	-0.294141	0.929655	
Tipo_Interes_Medio	0.974735	-0.474116	
Tasa_Paro_(%)	-0.019709	-0.571894	
Contratos_Indefinidos	-0.507466	0.822764	
Contratos_Temporales	0.186875	0.253944	
Poblacion	-0.640493	0.431957	
Cantidad_De_Extranjeros	-0.238827	0.512858	
Pib_Trimestral_Ajustado	-0.641555	0.747853	
Pib_Trimestral_No_Ajustado	-0.613126	0.722665	
Ipi_Variacion_Anual_Original	-0.239454	0.001944	
Ipi_Variacion_Anual_Corregida	-0.252022	0.017761	
Precio_M2_Vivienda	0.183712	0.653093	
Euribor_12_Meses	0.677100	0.015926	
Mro_Rate	0.628849	-0.041713	
Ti_Credito	0.799372	-0.125879	
Renta_Disponible_Bruta	-0.554828	0.606716	
Ipc_%_Variación_Anual	0.085500	0.316017	
Interes_Fijo_Hipotecas	0.951608	-0.531009	
Interes_Variable_Hipotecas	1.000000	-0.454372	
Afiliaciones_Ss	-0.454372	1.000000	
Duracion_Media_Hipoteca	0.050901	0.640637	
Numero_Compraventas	-0.087233	0.450785	
Valor_Compraventa	-0.150011	0.676177	
Confianza_Del_Consumidor	-0.191904	-0.129756	

	Duracion_Media_Hipoteca	Numero_Compraventas	\
Importe_Hipotecas	0.746574	0.768229	
Valor_Medio_Hipotecas	0.793670	0.535810	
Tipo_Interes_Medio	0.020786	-0.120291	
Tasa_Paro_(%)	-0.674780	-0.880105	
Contratos_Indefinidos	0.324100	-0.045953	
Contratos_Temporales	0.494347	0.687213	
Poblacion	-0.101752	-0.334111	
Cantidad_De_Extranjeros	0.232716	-0.280499	
Pib_Trimestral_Ajustado	0.145906	-0.054192	
Pib_Trimestral_No_Ajustado	0.111709	-0.000516	
Ipi_Variacion_Anual_Original	0.000737	0.368657	
Ipi_Variacion_Anual_Corregida	0.023746	0.403471	
Precio_M2_Vivienda	0.822461	0.332044	
Euribor_12_Meses	0.455817	0.352518	
Mro_Rate	0.421765	0.431793	

Ti_Credito	0.316043	0.191031
Renta_Disponible_Bruta	0.071447	-0.038089
Ipc_%_Variación_Anual	0.520953	0.574073
Interes_Fijo_Hipotecas	-0.085913	-0.304384
Interes_Variable_Hipotecas	0.050901	-0.087233
Afiliaciones_Ss	0.640637	0.450785
Duracion_Media_Hipoteca	1.000000	0.554129
Numeros_Compraventas	0.554129	1.000000
Valor_Compraventa	0.729017	0.937208
Confianza_Del_Consumidor	-0.273315	0.313941
Importe_Hipotecas	0.782869	0.062144
Valor_Medio_Hipotecas	0.761831	-0.169626
Tipo_Interes_Medio	-0.187478	-0.222398
Tasa_Paro_(%)	-0.873950	-0.232743
Contratos_Indefinidos	0.225139	-0.374532
Contratos_Temporales	0.614701	0.389491
Poblacion	-0.150372	-0.399179
Cantidad_De_Extranjeros	0.000182	-0.692205
Pib_Trimestral_Ajustado	0.154577	-0.073795
Pib_Trimestral_No_Ajustado	0.196619	-0.097641
Ipi_Variacion_Anual_Original	0.272215	0.543933
Ipi_Variacion_Anual_Corregida	0.308428	0.595106
Precio_M2_Vivienda	0.581715	-0.450602
Euribor_12_Meses	0.349632	-0.168635
Mro_Rate	0.394499	-0.107019
Ti_Credito	0.162773	-0.172817
Renta_Disponible_Bruta	0.154278	-0.225647
Ipc_%_Variación_Anual	0.577070	-0.115981
Interes_Fijo_Hipotecas	-0.333898	-0.273653
Interes_Variable_Hipotecas	-0.150011	-0.191904
Afiliaciones_Ss	0.676177	-0.129756
Duracion_Media_Hipoteca	0.729017	-0.273315
Numeros_Compraventas	0.937208	0.313941
Valor_Compraventa	1.000000	0.100132
Confianza_Del_Consumidor	0.100132	1.000000

[25 rows x 25 columns]

Resumen de las variables que están más correlacionadas (<-0.7 y >0.7) en función del método utilizado:

```
In [32]: vars_exp = iqr_winsorized_df.columns.drop(['Numero_Hipotecas', 'Fecha', 'Fecha_ordinal', 'Trimestre_simple', 'Añ

corr_pearson = iqr_winsorized_df[vars_exp].corr(method='pearson')

corr_spearman = iqr_winsorized_df[vars_exp].corr(method='spearman')

def get_strong_corrs(corr_matrix, threshold=0.7):
    strong_pairs = []
    for i in range(len(corr_matrix.columns)):
        for j in range(i+1, len(corr_matrix.columns)):
            val = corr_matrix.iloc[i, j]
            if abs(val) >= threshold:
                var1 = corr_matrix.index[i]
                var2 = corr_matrix.columns[j]
                strong_pairs.append((var1, var2, val))
    return strong_pairs

threshold = 0.7

strong_pearson = get_strong_corrs(corr_pearson, threshold)
strong_spearman = get_strong_corrs(corr_spearman, threshold)

print(f"Variables explicativas correlacionadas (|correlación| ≥ {threshold}) según Pearson:")
if strong_pearson:
    for v1, v2, val in strong_pearson:
        print(f" - {v1} y {v2}: correlación = {val:.2f}")
else:
    print(" - Ninguna pareja con correlación fuerte.")

print(f"\nVariables explicativas correlacionadas (|correlación| ≥ {threshold}) según Spearman:")
if strong_spearman:
    for v1, v2, val in strong_spearman:
        print(f" - {v1} y {v2}: correlación = {val:.2f}")
else:
    print(" - Ninguna pareja con correlación fuerte.")
```

Variables explicativas correlacionadas ($|correlación| \geq 0.7$) según Pearson:

- Importe_Hipotecas y Tasa_Paro_(%): correlación = -0.87
- Importe_Hipotecas y Contratos_Temporales: correlación = 0.82
- Importe_Hipotecas y Mro_Rate: correlación = 0.77
- Importe_Hipotecas y Duracion_Media_Hipoteca: correlación = 0.78
- Importe_Hipotecas y Numero_Compraventas: correlación = 0.80
- Importe_Hipotecas y Valor_Compraventa: correlación = 0.80
- Valor_Medio_Hipotecas y Precio_M2_Vivienda: correlación = 0.74
- Valor_Medio_Hipotecas y Afiliaciones_Ss: correlación = 0.91
- Valor_Medio_Hipotecas y Duracion_Media_Hipoteca: correlación = 0.76
- Valor_Medio_Hipotecas y Valor_Compraventa: correlación = 0.75
- Tipo_Interes_Medio y Ti_Credito: correlación = 0.85
- Tipo_Interes_Medio y Interes_Fijo_Hipotecas: correlación = 0.93
- Tipo_Interes_Medio y Interes_Variable_Hipotecas: correlación = 0.97
- Tasa_Paro_(%) y Contratos_Temporales: correlación = -0.77
- Tasa_Paro_(%) y Numero_Compraventas: correlación = -0.87
- Tasa_Paro_(%) y Valor_Compraventa: correlación = -0.86
- Contratos_Indefinidos y Poblacion: correlación = 0.82
- Contratos_Indefinidos y Cantidad_De_Extranjeros: correlación = 0.89
- Contratos_Indefinidos y Pib_Trimestral_Ajustado: correlación = 0.95
- Contratos_Indefinidos y Pib_Trimestral_No_Ajustado: correlación = 0.93
- Contratos_Indefinidos y Renta_Disponible_Bruta: correlación = 0.83
- Contratos_Indefinidos y Afiliaciones_Ss: correlación = 0.84
- Contratos_Temporales y Poblacion: correlación = -0.76
- Contratos_Temporales y Numero_Compraventas: correlación = 0.74
- Poblacion y Cantidad_De_Extranjeros: correlación = 0.87
- Poblacion y Pib_Trimestral_Ajustado: correlación = 0.84
- Poblacion y Pib_Trimestral_No_Ajustado: correlación = 0.83
- Poblacion y Renta_Disponible_Bruta: correlación = 0.77
- Cantidad_De_Extranjeros y Pib_Trimestral_Ajustado: correlación = 0.80
- Cantidad_De_Extranjeros y Pib_Trimestral_No_Ajustado: correlación = 0.78
- Cantidad_De_Extranjeros y Renta_Disponible_Bruta: correlación = 0.73
- Pib_Trimestral_Ajustado y Pib_Trimestral_No_Ajustado: correlación = 0.98
- Pib_Trimestral_Ajustado y Renta_Disponible_Bruta: correlación = 0.84
- Pib_Trimestral_Ajustado y Afiliaciones_Ss: correlación = 0.78
- Pib_Trimestral_No_Ajustado y Renta_Disponible_Bruta: correlación = 0.91
- Pib_Trimestral_No_Ajustado y Afiliaciones_Ss: correlación = 0.77
- Ipi_Variacion_Annual_Original y Ipi_Variacion_Annual_Corregida: correlación = 0.94
- Precio_M2_Vivienda y Duracion_Media_Hipoteca: correlación = 0.84
- Euribor_12_Meses y Mro_Rate: correlación = 0.91
- Euribor_12_Meses y Ti_Credito: correlación = 0.93
- Mro_Rate y Ti_Credito: correlación = 0.80
- Ti_Credito y Interes_Variable_Hipotecas: correlación = 0.82
- Interes_Fijo_Hipotecas y Interes_Variable_Hipotecas: correlación = 0.95
- Duracion_Media_Hipoteca y Valor_Compraventa: correlación = 0.71
- Numero_Compraventas y Valor_Compraventa: correlación = 0.93

Variables explicativas correlacionadas ($|correlación| \geq 0.7$) según Spearman:

- Importe_Hipotecas y Tasa_Paro_(%): correlación = -0.87
- Importe_Hipotecas y Contratos_Temporales: correlación = 0.75
- Importe_Hipotecas y Duracion_Media_Hipoteca: correlación = 0.75
- Importe_Hipotecas y Numero_Compraventas: correlación = 0.77
- Importe_Hipotecas y Valor_Compraventa: correlación = 0.78
- Valor_Medio_Hipotecas y Precio_M2_Vivienda: correlación = 0.76
- Valor_Medio_Hipotecas y Afiliaciones_Ss: correlación = 0.93
- Valor_Medio_Hipotecas y Duracion_Media_Hipoteca: correlación = 0.79
- Valor_Medio_Hipotecas y Valor_Compraventa: correlación = 0.76
- Tipo_Interes_Medio y Ti_Credito: correlación = 0.82
- Tipo_Interes_Medio y Interes_Fijo_Hipotecas: correlación = 0.94
- Tipo_Interes_Medio y Interes_Variable_Hipotecas: correlación = 0.97
- Tasa_Paro_(%) y Contratos_Temporales: correlación = -0.77
- Tasa_Paro_(%) y Numero_Compraventas: correlación = -0.88
- Tasa_Paro_(%) y Valor_Compraventa: correlación = -0.87
- Contratos_Indefinidos y Poblacion: correlación = 0.76
- Contratos_Indefinidos y Cantidad_De_Extranjeros: correlación = 0.84
- Contratos_Indefinidos y Pib_Trimestral_Ajustado: correlación = 0.89
- Contratos_Indefinidos y Pib_Trimestral_No_Ajustado: correlación = 0.86
- Contratos_Indefinidos y Renta_Disponible_Bruta: correlación = 0.78
- Contratos_Indefinidos y Afiliaciones_Ss: correlación = 0.82
- Poblacion y Cantidad_De_Extranjeros: correlación = 0.76
- Poblacion y Pib_Trimestral_Ajustado: correlación = 0.73
- Poblacion y Pib_Trimestral_No_Ajustado: correlación = 0.73
- Poblacion y Renta_Disponible_Bruta: correlación = 0.72
- Pib_Trimestral_Ajustado y Pib_Trimestral_No_Ajustado: correlación = 0.95
- Pib_Trimestral_Ajustado y Renta_Disponible_Bruta: correlación = 0.77
- Pib_Trimestral_Ajustado y Afiliaciones_Ss: correlación = 0.75
- Pib_Trimestral_No_Ajustado y Renta_Disponible_Bruta: correlación = 0.89
- Pib_Trimestral_No_Ajustado y Afiliaciones_Ss: correlación = 0.72
- Ipi_Variacion_Annual_Original y Ipi_Variacion_Annual_Corregida: correlación = 0.87
- Precio_M2_Vivienda y Duracion_Media_Hipoteca: correlación = 0.82

- Euribor_12_Meses y Mro_Rate: correlación = 0.95
 - Euribor_12_Meses y Ti_Credito: correlación = 0.94
 - Mro_Rate y Ti_Credito: correlación = 0.88
 - Ti_Credito y Interes_Variable_Hipotecas: correlación = 0.80
 - Interes_Fijo_Hipotecas y Interes_Variable_Hipotecas: correlación = 0.95
 - Duracion_Media_Hipoteca y Valor_Compraventa: correlación = 0.73
 - Numero_Compraventas y Valor_Compraventa: correlación = 0.94

Gráficamente:

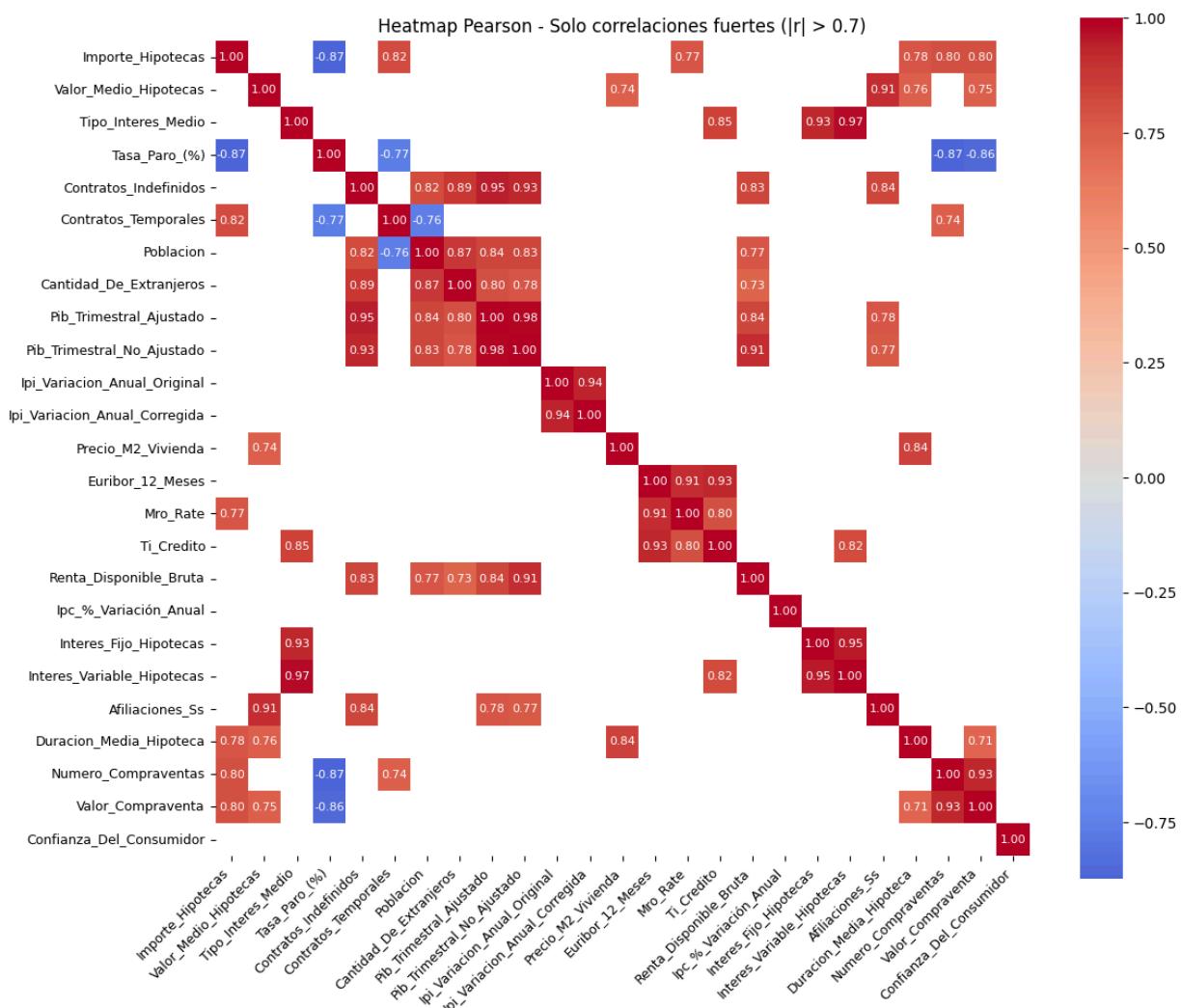
```
In [33]: corr_pearson_strong = corr_pearson.copy()
corr_spearman_strong = corr_spearman.copy()

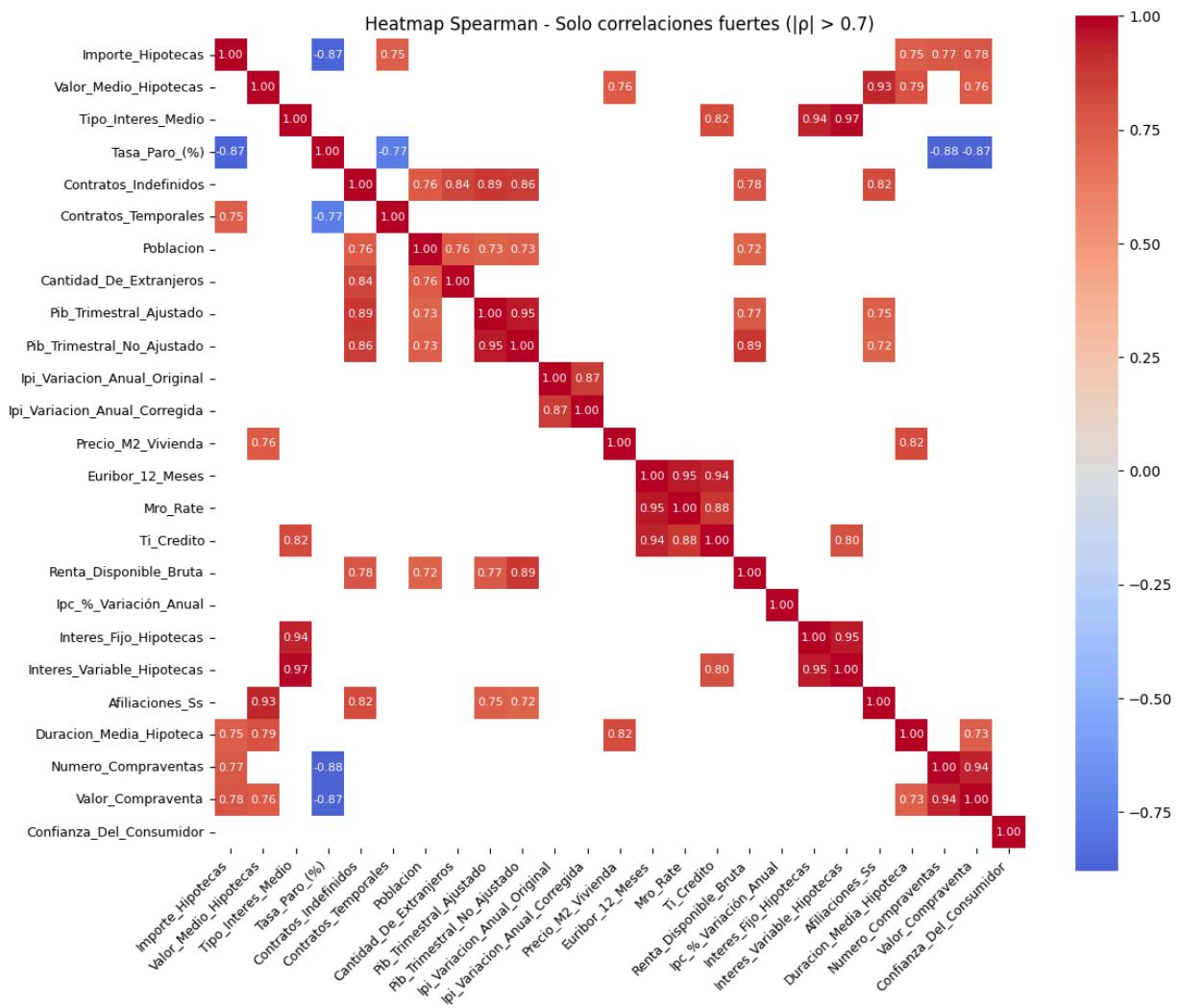
# Pearson
mask_pearson = (corr_pearson_strong.abs() < 0.7) | (corr_pearson_strong.isna())

plt.figure(figsize=(12, 10))
sns.heatmap(corr_pearson_strong, mask=mask_pearson, annot=True, cmap='coolwarm', center=0, square=True, fmt=".2f")
plt.title('Heatmap Pearson - Solo correlaciones fuertes (|r| > 0.7)')
plt.xticks(rotation=45, ha='right', fontsize=9)
plt.yticks(rotation=0, fontsize=9)
plt.tight_layout()
plt.show()

# Spearman
mask_spearman = (corr_spearman_strong.abs() < 0.7) | (corr_spearman_strong.isna())

plt.figure(figsize=(12, 10))
sns.heatmap(corr_spearman_strong, mask=mask_spearman, annot=True, cmap='coolwarm', center=0, square=True, fmt=".2f")
plt.title('Heatmap Spearman - Solo correlaciones fuertes (|ρ| > 0.7)')
plt.xticks(rotation=45, ha='right', fontsize=9)
plt.yticks(rotation=0, fontsize=9)
plt.tight_layout()
plt.show()
```





Correlación con la variable objetivo

A continuación, vamos a estudiar la correlación de todas las variables explicativas con la variable objetivo. De nuevo, usaremos tanto el método de Pearson como el método de Spearman para ver las diferencias:

```
In [34]: # Lista de variables numéricas (excepto la variable objetivo)
vars_exp = iqr_winsorized_df.select_dtypes(include='number').columns.drop(['Numero_Hipotecas', 'Fecha_ordinal'])

# Correlación de cada variable con la variable objetivo
corr_pearson_y = iqr_winsorized_df[vars_exp].corrwith(iqr_winsorized_df['Numero_Hipotecas'], method='pearson')
corr_spearman_y = iqr_winsorized_df[vars_exp].corrwith(iqr_winsorized_df['Numero_Hipotecas'], method='spearman')

# Mostrar resultados
print("📊 Correlación Pearson con 'Numero_Hipotecas':\n")
print(corr_pearson_y.sort_values(ascending=False))

print("\n📊 Correlación Spearman con 'Numero_Hipotecas':\n")
print(corr_spearman_y.sort_values(ascending=False))

# Comparación Lado a Lado
correlation_comparison = pd.DataFrame({
    'Pearson': corr_pearson_y,
    'Spearman': corr_spearman_y
}).sort_values(by='Pearson', ascending=False)

print("\n📋 Comparación de correlaciones con 'Numero_Hipotecas':\n")
print(correlation_comparison)
```

📊 Correlación Pearson con 'Numero_Hipotecas':

```
Importe_Hipotecas      0.944256
Contratos_Temporales  0.840968
Mro_Rate              0.815825
Numero_Compraventas  0.793368
Valor_Compraventa    0.709495
Euribor_12_Meses      0.677857
Duracion_Media_Hipoteca  0.675846
Ti_Credito            0.576598
Precio_M2_Vivienda    0.472683
Interes_Variable_Hipotecas  0.418711
Ipc_%_Variación_Anual 0.402319
Tipo_Interes_Medio    0.368688
Valor_Medio_Hipotecas 0.330735
Interes_Fijo_Hipotecas 0.233259
Confianza_Del_Consumidor  0.217765
Ipi_Variacion_Anual_Corregida  0.184646
Ipi_Variacion_Anual_Original  0.178633
Afiliaciones_Ss        0.080779
Contratos_Indefinidos -0.393566
Renta_Disponible_Bruta -0.495540
Pib_Trimestral_Ajustado -0.499242
Pib_Trimestral_No_Ajustado -0.507156
Cantidad_De_Extranjeros -0.527495
Tasa_Paro_(%)          -0.773247
Poblacion              -0.828708
dtype: float64
```

📊 Correlación Spearman con 'Numero_Hipotecas':

```
Importe_Hipotecas      0.990361
Numero_Compraventas  0.752030
Contratos_Temporales  0.748122
Valor_Compraventa    0.745226
Duracion_Media_Hipoteca  0.701300
Mro_Rate              0.654031
Precio_M2_Vivienda    0.615086
Euribor_12_Meses      0.610259
Ipc_%_Variación_Anual 0.585849
Valor_Medio_Hipotecas 0.486868
Ti_Credito            0.468232
Afiliaciones_Ss        0.332267
Interes_Variable_Hipotecas  0.265510
Tipo_Interes_Medio    0.220079
Ipi_Variacion_Anual_Corregida  0.134453
Ipi_Variacion_Anual_Original  0.121890
Confianza_Del_Consumidor  0.088164
Interes_Fijo_Hipotecas  0.058542
Contratos_Indefinidos -0.085983
Cantidad_De_Extranjeros -0.163621
Pib_Trimestral_Ajustado -0.258850
Pib_Trimestral_No_Ajustado -0.290959
Renta_Disponible_Bruta -0.318347
Poblacion              -0.520924
Tasa_Paro_(%)          -0.833107
dtype: float64
```

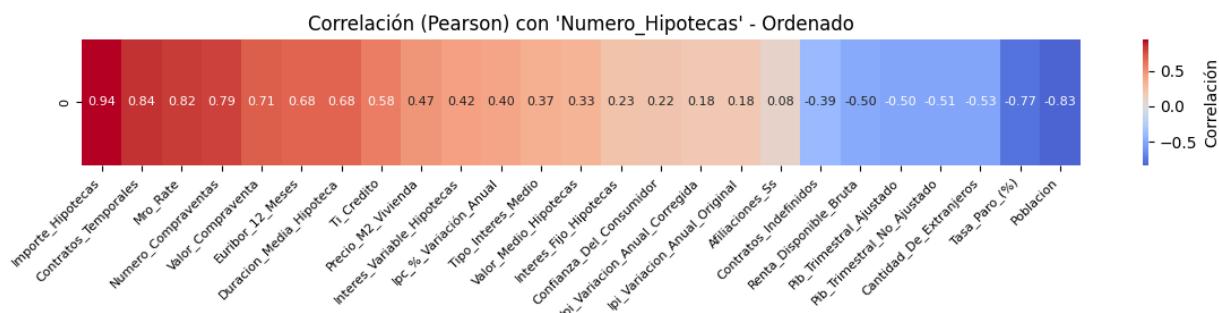
📋 Comparación de correlaciones con 'Numero_Hipotecas':

	Pearson	Spearman
Importe_Hipotecas	0.944256	0.990361
Contratos_Temporales	0.840968	0.748122
Mro_Rate	0.815825	0.654031
Numero_Compraventas	0.793368	0.752030
Valor_Compraventa	0.709495	0.745226
Euribor_12_Meses	0.677857	0.610259
Duracion_Media_Hipoteca	0.675846	0.701300
Ti_Credito	0.576598	0.468232
Precio_M2_Vivienda	0.472683	0.615086
Interes_Variable_Hipotecas	0.418711	0.265510
Ipc_%_Variación_Anual	0.402319	0.585849
Tipo_Interes_Medio	0.368688	0.220079
Valor_Medio_Hipotecas	0.330735	0.486868
Interes_Fijo_Hipotecas	0.233259	0.058542
Confianza_Del_Consumidor	0.217765	0.088164
Ipi_Variacion_Anual_Corregida	0.184646	0.134453
Ipi_Variacion_Anual_Original	0.178633	0.121890
Afiliaciones_Ss	0.080779	0.332267
Contratos_Indefinidos	-0.393566	-0.085983

Renta_Disponible_Bruta	-0.495540	-0.318347
Pib_Trimestral_Ajustado	-0.499242	-0.258850
Pib_Trimestral_No_Ajustado	-0.507156	-0.290959
Cantidad_De_Extranjeros	-0.527495	-0.163621
Tasa_Paro_(%)	-0.773247	-0.833107
Poblacion	-0.828708	-0.520924

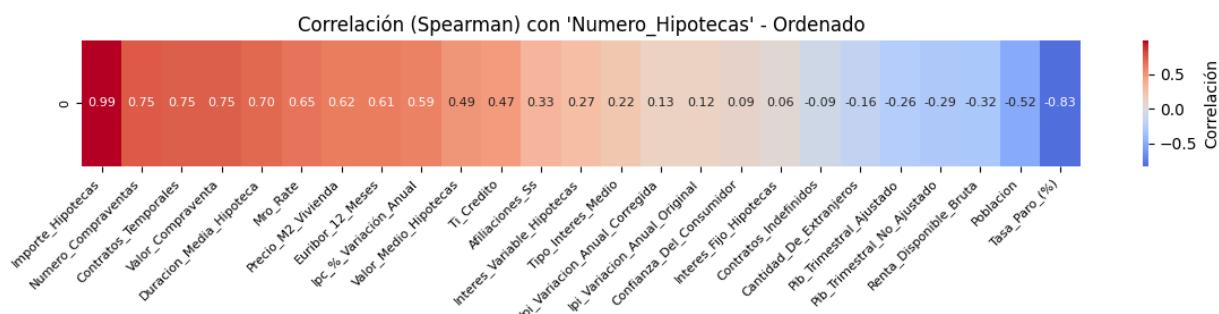
Vemos con un heatmap la correlación mediante Pearson con la variable objetivo:

```
In [35]: corr_pearson_y_sorted = corr_pearson_y.sort_values(ascending=False)
plt.figure(figsize=(12, 3))
sns.heatmap(corr_pearson_y_sorted.to_frame().T, cmap='coolwarm', center=0,
            annot=True, fmt=".2f", cbar_kws={'label': 'Correlación'},
            annot_kws={"size": 8})
plt.title("Correlación (Pearson) con 'Numero_Hipotecas' - Ordenado", fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=8)
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
```



Vemos con un heatmap la correlación mediante Spearman con la variable objetivo:

```
In [36]: corr_spearman_y_sorted = corr_spearman_y.sort_values(ascending=False)
plt.figure(figsize=(12, 3))
sns.heatmap(corr_spearman_y_sorted.to_frame().T, cmap='coolwarm', center=0,
            annot=True, fmt=".2f", cbar_kws={'label': 'Correlación'},
            annot_kws={"size": 8})
plt.title("Correlación (Spearman) con 'Numero_Hipotecas' - Ordenado", fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=8)
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
```



Resultados y Eliminación de variables correlacionadas

Cuando dos o más variables están fuertemente correlacionadas, incluirlas todas en una regresión u otros modelos temporales puede provocar multicolinealidad, lo que puede afectar a la interpretación y a los coeficientes.

Por tanto, vamos a crear un nuevo Dataframe con las variables seleccionadas tras el análisis de multicolinealidad. No obstante, en el próximo notebook haremos uso de ambos dataframes (antes de la eliminación y después) para comprobar cómo afecta a los modelos y regresiones.

- **Importe_Hipotecas:** Alta correlación, directa y lógica (0.94 Pearson y 0.99 Spearman). Además, presenta una alta correlación con otras variables explicativas que no son tan predictoras de la variable objetivo.
- **Número_Compraventas:** Presenta una correlación moderada (0.47 Pearson y 0.5 Spearman), pero su relación directa con la variable objetivo y su relación teórica hace que decidimos mantenerla para los modelos.

- **Precio_M2_Vivienda:** Presenta una correlación moderada (0.61 Spearman), pero es un índice estructural del mercado inmobiliario que es útil para la predicción. Lo mantenemos al eliminar variables similares.
- **Valor_Compraventa:** Alta correlación y sentido lógico (0.70 Pearson y 0.74 Spearman); más compraventas implican más necesidad de financiación hipotecaria.
- **Contratos_Temporales:** Decidimos mantenerla por su alta correlación (0.84 Pearson y 0.74 Spearman), aunque puede ser dudosa su causalidad directa.
- **Ipc_%_Variacion_Anual:** La correlación no es muy elevada (0.58 Spearman), aunque puede ser útil mantenerla como reflejo del entorno económico general. Tiene una utilidad contextual.
- **Duracion_Media_Hipoteca:** Buena correlación (0.67 Pearson y 0.70 Spearman); representa la evolución de la financiación. Una mayor duración puede estar ligada a un mayor volumen de hipotecas.
- **Tasa_Paro_(%):** Correlación fuerte y esperada (-0.77 Pearson y 0.83 Spearman); a mayor paro, menor demanda hipotecaria.

```
In [37]: columnas_seleccionadas = [
    "Fecha",
    "Numero_Hipotecas",
    "Importe_Hipotecas",
    "Numero_Compraventas",
    "Precio_M2_Vivienda",
    "Contratos_Temporales",
    "Ipc_%_Variación_Anual",
    "Duracion_Media_Hipoteca",
    "Tasa_Paro_(%)"
]

df_no_corr = iqr_winsorized_df[columnas_seleccionadas].copy()
df_no_corr.head()
```

	Fecha	Numero_Hipotecas	Importe_Hipotecas	Numero_Compraventas	Precio_M2_Vivienda	Contratos_Temporales	Ipc_%
0	2003-03-31	263600	23.908609	183908.527902	1230.3	4370.2	
1	2003-06-30	247071	23.876021	185520.576206	1309.6	4550.9	
2	2003-09-30	236977	23.900416	187146.754903	1344.9	4652.2	
3	2003-12-31	241791	23.926874	188787.187852	1380.3	4687.5	
4	2004-03-31	283170	24.125194	190442.000000	1456.2	4620.7	

Descarga de Excels

```
In [41]: from openpyxl import load_workbook
from openpyxl.utils import get_column_letter
from openpyxl.styles import numbers
from openpyxl.styles import PatternFill

# Guardar el DataFrame en Excel sin índice
nombre_archivo = "df_no_corr.xlsx"
df_no_corr.to_excel(nombre_archivo, sheet_name="variables", index=False)

# Ajustar el ancho de las columnas con openpyxl
# Cargar el archivo
wb = load_workbook(nombre_archivo)
ws = wb["variables"]

# Ajustar el ancho de cada columna según el contenido más Largo
for col_idx, column_cells in enumerate(ws.iter_cols(min_row=1, max_row=1), 1):
    max_length = 0
    for cell in column_cells:
        try:
            max_length = max(max_length, len(str(cell.value)))
        except:
            pass
```

```

adjusted_width = max_length + 2 # Un poco de margen
ws.column_dimensions[get_column_letter(col_idx)].width = adjusted_width

# Ajustar el ancho de la columna Fecha, al ser la primera es la "A"
ws.column_dimensions["A"].width = 20

# Aplicar formato de fecha corta a la columna A, si no, por defecto, Excel nos pone la fecha Larga incluyendo la
for cell in ws["A"][1:]: # Saltamos la cabecera
    cell.number_format = 'dd/mm/yyyy' # Este formato Excel lo traduce como "Fecha corta"

# Definir los colores
fill_header = PatternFill(start_color="A9D0F5", end_color="A9D0F5", fill_type="solid") # Azul claro para encabezado
fill_gray = PatternFill(start_color="D9D9D9", end_color="D9D9D9", fill_type="solid") # Gris claro
fill_white = PatternFill(start_color="FFFFFF", end_color="FFFFFF", fill_type="solid")

# Aplicar color al encabezado (primera fila)
for cell in ws[1]:
    cell.fill = fill_header

# Aplicar color por bloques de 4 filas
for i, row in enumerate(ws.iter_rows(min_row=2), start=0): # min_row=2 para saltar encabezado
    fill = fill_gray if (i // 4) % 2 == 0 else fill_white
    for cell in row:
        cell.fill = fill

# Guardar el archivo actualizado
wb.save(nombre_archivo)
print("Archivo df_no_corr.xlsx guardado y formateado correctamente")

```

Archivo df_no_corr.xlsx guardado y formateado correctamente

```

In [42]: # Guardar el DataFrame en Excel sin índice
nombre_archivo = "iqr_winsorized_df.xlsx"
iqr_winsorized_df.to_excel(nombre_archivo, sheet_name="variables", index=False)

# Ajustar el ancho de las columnas con openpyxl
# Cargar el archivo
wb = load_workbook(nombre_archivo)
ws = wb["variables"]

# Ajustar el ancho de cada columna según el contenido más largo
for col_idx, column_cells in enumerate(ws.iter_cols(min_row=1, max_row=1), 1):
    max_length = 0
    for cell in column_cells:
        try:
            max_length = max(max_length, len(str(cell.value)))
        except:
            pass
    adjusted_width = max_length + 2 # Un poco de margen
    ws.column_dimensions[get_column_letter(col_idx)].width = adjusted_width

# Ajustar el ancho de la columna Fecha, al ser la primera es la "A"
ws.column_dimensions["A"].width = 20

# Aplicar formato de fecha corta a la columna A, si no, por defecto, Excel nos pone la fecha Larga incluyendo la
for cell in ws["A"][1:]: # Saltamos la cabecera
    cell.number_format = 'dd/mm/yyyy' # Este formato Excel lo traduce como "Fecha corta"

# Definir los colores
fill_header = PatternFill(start_color="A9D0F5", end_color="A9D0F5", fill_type="solid") # Azul claro para encabezado
fill_gray = PatternFill(start_color="D9D9D9", end_color="D9D9D9", fill_type="solid") # Gris claro
fill_white = PatternFill(start_color="FFFFFF", end_color="FFFFFF", fill_type="solid")

# Aplicar color al encabezado (primera fila)
for cell in ws[1]:
    cell.fill = fill_header

# Aplicar color por bloques de 4 filas
for i, row in enumerate(ws.iter_rows(min_row=2), start=0): # min_row=2 para saltar encabezado
    fill = fill_gray if (i // 4) % 2 == 0 else fill_white
    for cell in row:
        cell.fill = fill

# Guardar el archivo actualizado
wb.save(nombre_archivo)
print("Archivo iqr_winsorized_df.xlsx guardado y formateado correctamente")

```

Archivo iqr_winsorized_df.xlsx guardado y formateado correctamente

In []:

MODELOS MULTIVARIANTES PARA LA PREDICCIÓN DE LA DEMANDA HIPOTECARIA

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from statsmodels.tsa.stattools import adfuller
import numpy as np
from xgboost import XGBRegressor
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.stats.diagnostic import het_breusvhpagan
from statsmodels.stats.stattools import durbin_watson
from scipy.stats import shapiro
from statsmodels.graphics.regressionplots import plot_partregress_grid
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.diagnostic import linear_reset
from statsmodels.graphics.regressionplots import plot_partregress
```

1. REGRESIÓN LINEAL MÚLTIPLE

```
In [2]: # Incorporamos Los dos dataframes que vamos a emplear en el modelo:
df_no_corr = pd.read_excel("df_no_corr.xlsx")
iqr_winsorized_df = pd.read_excel("iqr_winsorized_df.xlsx")
```

```
In [3]: df_no_corr.head()
```

```
Out[3]:
```

	Fecha	Numero_Hipotecas	Importe_Hipotecas	Numero_Compraventas	Precio_M2_Vivienda	Contratos_Temporales	Ipc_%
0	2003-03-31	263600	23.908609	183908.527902	1230.3	4370.2	
1	2003-06-30	247071	23.876021	185520.576206	1309.6	4550.9	
2	2003-09-30	236977	23.900416	187146.754903	1344.9	4652.2	
3	2003-12-31	241791	23.926874	188787.187852	1380.3	4687.5	
4	2004-03-31	283170	24.125194	190442.000000	1456.2	4620.7	

```
In [4]: # Nos aseguramos de que la columna 'Fecha' esté en formato datetime y como índice
df_no_corr["Fecha"] = pd.to_datetime(df_no_corr["Fecha"])
df_no_corr.set_index("Fecha", inplace=True)
```

```
In [5]: iqr_winsorized_df = iqr_winsorized_df.drop(['Fecha_ordinal', 'Trimestre_simple', 'Año', 'Periodo'], axis=1)
iqr_winsorized_df.head()
```

Out[5]:

	Fecha	Numero_Hipotecas	Importe_Hipotecas	Valor_Medio_Hipotecas	Tipo_Interes_Medio	Tasa_Paro_(%)	Contratos_Inde
0	2003-03-31	263600	23.908609	91713.17	4.693333	11.99	9.
1	2003-06-30	247071	23.876021	94711.40	4.453333	11.28	9.
2	2003-09-30	236977	23.900416	101184.20	4.213333	11.30	9.
3	2003-12-31	241791	23.926874	101828.44	3.883333	11.37	9.
4	2004-03-31	283170	24.125194	106020.83	3.760000	11.50	9.

5 rows × 27 columns

In [6]:

```
# Nos aseguramos de que La columna 'Fecha' esté en formato datetime y como índice
iqr_winsorized_df["Fecha"] = pd.to_datetime(iqr_winsorized_df["Fecha"])
iqr_winsorized_df.set_index("Fecha", inplace=True)
```

Paso 1: Ajuste del modelo y obtención de coeficientes para df_no_corr y iqr_winsorized_df

df_no_corr

In [7]:

```
df = df_no_corr.copy()

# Ajuste del modelo
X = df.drop(columns=['Numero_Hipotecas'])
y = df['Numero_Hipotecas']
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

# Coeficientes
print("--- Coeficientes del modelo ---")
print(model.params)
```

```
--- Coeficientes del modelo ---
const              -4.134786e+06
Importe_Hipotecas      1.602816e+05
Numero_Compraventas    3.239932e-01
Precio_M2_Vivienda      -1.004606e+02
Contratos_Temporales     2.082219e+01
Ipc_%_Variación_Anual   -2.542549e+03
Duracion_Media_Hipoteca 1.405699e+05
Tasa_Paro_(%)            7.291619e+03
dtype: float64
```

iqr_winsorized_df

In [8]:

```
df2 = iqr_winsorized_df.copy()

# Ajuste del modelo
X2 = df2.drop(columns=['Numero_Hipotecas'])
y2 = df2['Numero_Hipotecas']
X2 = sm.add_constant(X2)

model2 = sm.OLS(y2, X2).fit()

# Coeficientes
print("--- Coeficientes del modelo ---")
print(model2.params)
```

```

--- Coeficientes del modelo ---
const 1.765667e+06
Importe_Hipotecas 7.346631e+04
Valor_Medio_Hipotecas 1.903861e-01
Tipo_Interes_Medio -1.767313e+04
Tasa_Paro_(%) 3.875321e+02
Contratos_Indefinidos -1.029158e+04
Contratos_Temporales -1.977115e+01
Poblacion -7.150544e-02
Cantidad_De_Extranjeros 3.379368e-02
Pib_Trimestral_Ajustado 3.909575e-01
Pib_Trimestral_No_Ajustado -4.977535e-01
Ipi_Variacion_Anual_Original 1.841756e+03
Ipi_Variacion_Anual_Corregida -1.580079e+03
Precio_M2_Vivienda -3.216741e+01
Euribor_12_Meses -6.819924e+03
Mro_Rate 3.942415e+03
Ti_Credito 1.357752e+04
Renta_Disponible_Bruta 1.154299e-01
Ipc_%_Variación_Anual -4.488855e+02
Interes_Fijo_Hipotecas 1.350438e+03
Interes_Variable_Hipotecas 5.168192e+03
Afiliaciones_Ss 1.937093e-03
Duracion_Media_Hipoteca -3.264615e+04
Numero_Compraventas -2.301852e-01
Valor_Compraventa 2.305097e-06
Confianza_Del_Consumidor 7.544087e+01
dtype: float64

```

Paso 2: Validación de supuestos

A) Linealidad de los parámetros

`df_no_corr`

```

In [9]: reset = linear_reset(model, power=2, use_f=True)
print("--- Ramsey RESET test ---")
print(f"F-stat: {reset.fvalue:.4f}, p-value: {reset.pvalue:.4f}")

if reset.pvalue < 0.05:
    print("⚠ Se rechaza H0: posible falta de linealidad o forma funcional incorrecta.")
else:
    print("✅ No se rechaza H0: no hay indicios de falta de linealidad o forma funcional incorrecta.")

--- Ramsey RESET test ---
F-stat: 438.8288, p-value: 0.0000
⚠ Se rechaza H0: posible falta de linealidad o forma funcional incorrecta.

```

```

In [10]: df_temp = df_no_corr.rename(
    columns=lambda x: x.replace(" ", "_").replace("(", "").replace(")", "").replace("%", "pct")
)

independent_vars_temp = [col for col in df_temp.columns if col != "Numero_Hipotecas"]

fig, axes = plt.subplots(len(independent_vars_temp), 1, figsize=(8, 4 * len(independent_vars_temp)))

if len(independent_vars_temp) == 1:
    axes = [axes]

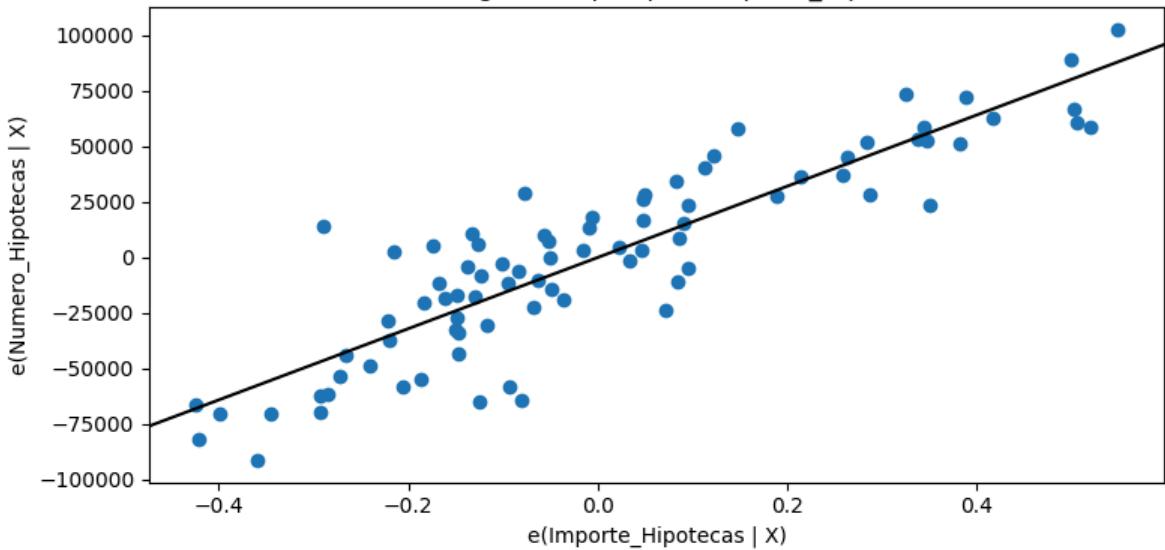
for ax, var in zip(axes, independent_vars_temp):
    # Generamos partial regression plot
    plot_partregress(endog="Numero_Hipotecas",
                      exog_i=var,
                      exog_others=[v for v in independent_vars_temp if v != var],
                      data=df_temp, ax=ax)
    ax.set_title(f'Partial regression plot para {var}')

    # Quitar las etiquetas de puntos (textos)
    for txt in ax.texts:
        txt.set_visible(False)

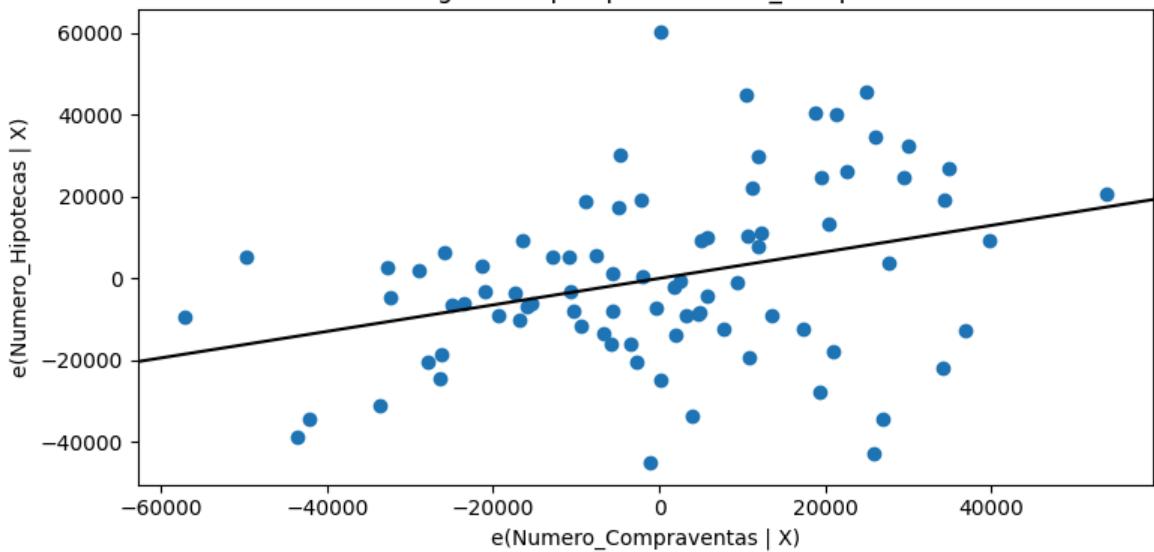
plt.tight_layout()
plt.show()

```

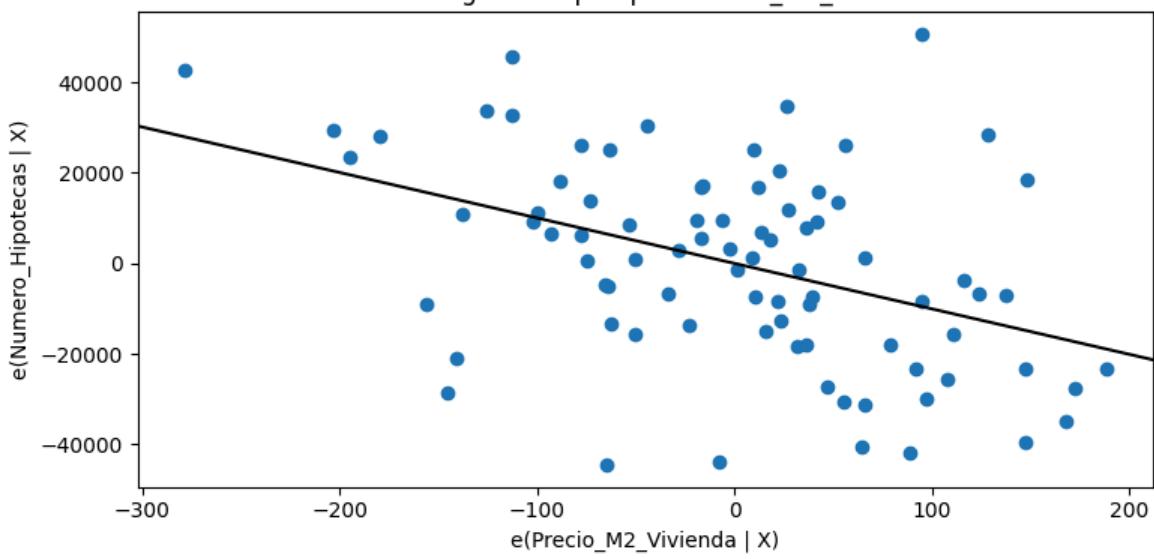
Partial regression plot para Importe_Hipotecas



Partial regression plot para Numero_Compraventas

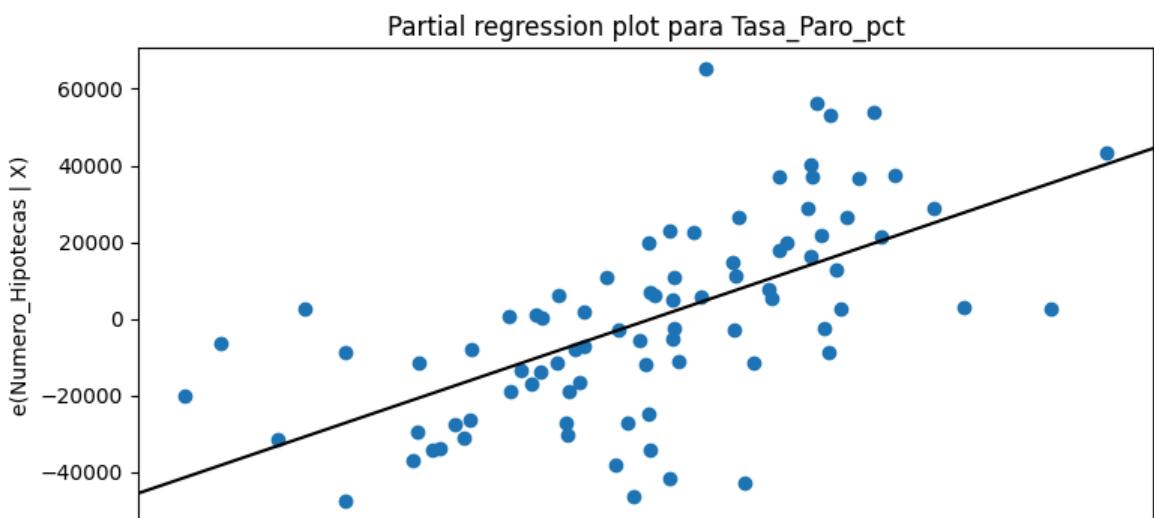
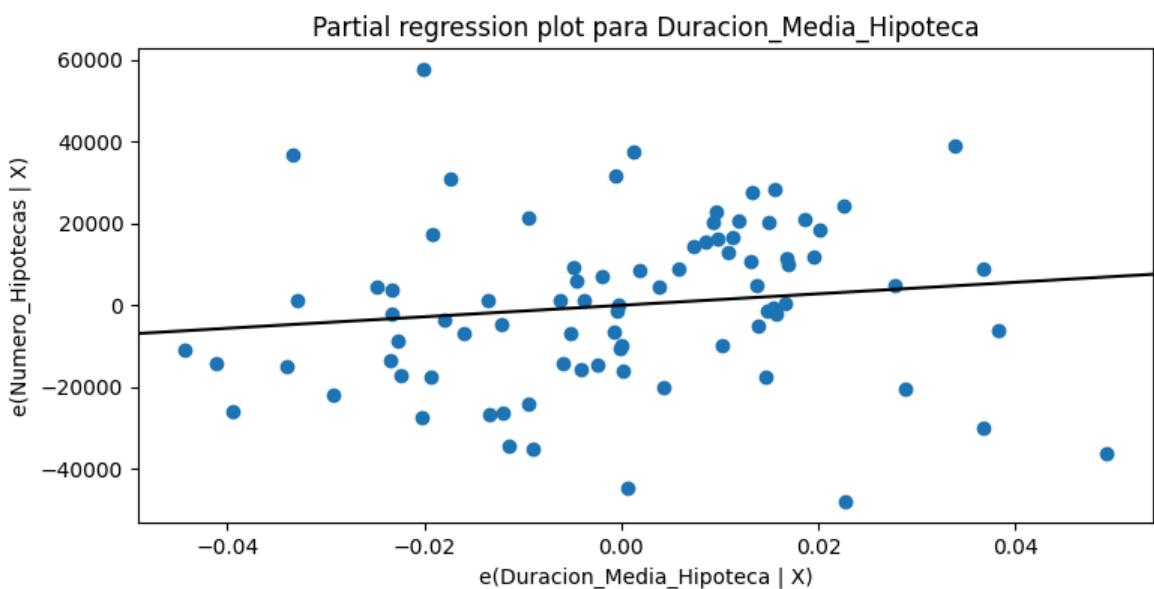
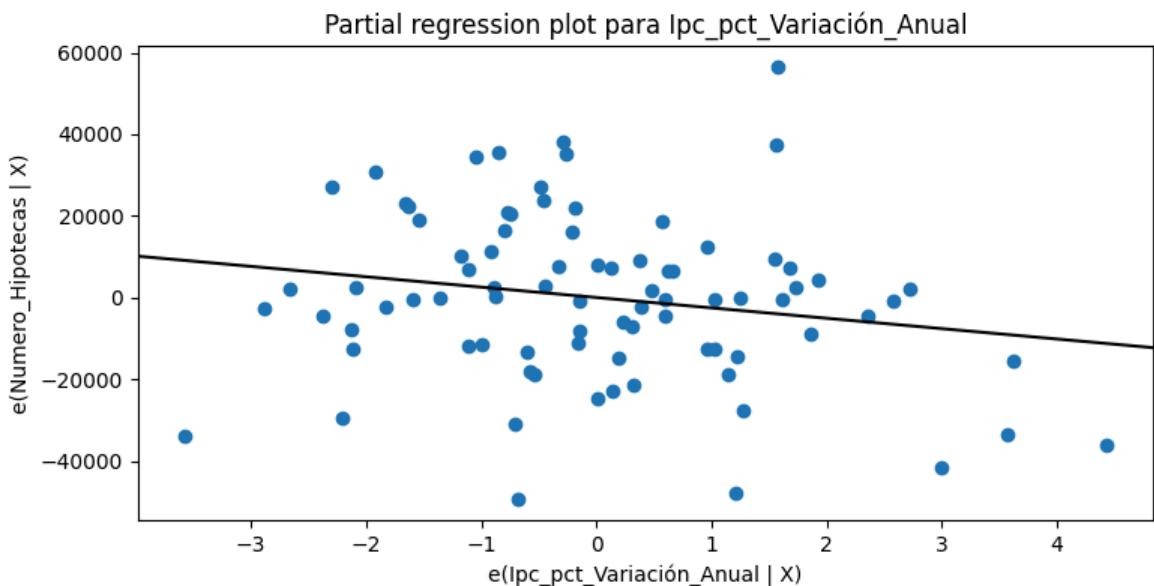
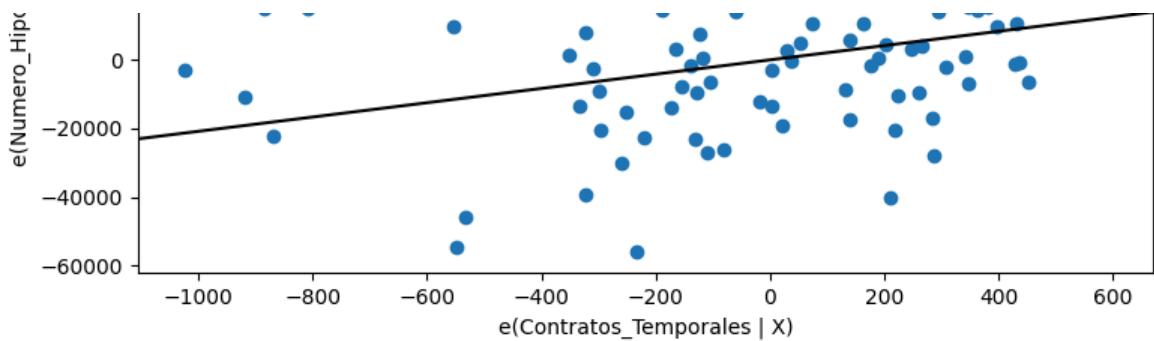


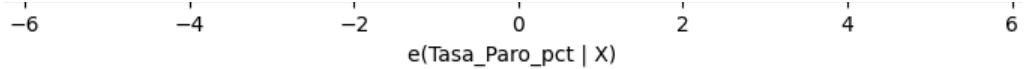
Partial regression plot para Precio_M2_Vivienda



Partial regression plot para Contratos_Temporales







iqr_winsorized_df

```
In [11]: reset = linear_reset(model2, power=2, use_f=True)
print("--- Ramsey RESET test ---")
print(f" F-stat: {reset.fvalue:.4f}, p-value: {reset.pvalue:.4f}")

if reset.pvalue < 0.05:
    print("⚠️ Se rechaza H0: posible falta de linealidad o forma funcional incorrecta.")
else:
    print("✅ No se rechaza H0: no hay indicios de falta de linealidad o forma funcional incorrecta.")

--- Ramsey RESET test ---
F-stat: 183.7115, p-value: 0.0000
⚠️ Se rechaza H0: posible falta de linealidad o forma funcional incorrecta.

In [12]: df_temp2 = iqr_winsorized_df.rename(
    columns=lambda x: x.replace(" ", "_").replace("(", "").replace(")", "").replace("%", "pct")
)

independent_vars_temp2 = [col for col in df_temp2.columns if col != "Numero_Hipotecas"]

fig, axes = plt.subplots(len(independent_vars_temp2), 1, figsize=(8, 4 * len(independent_vars_temp2)))

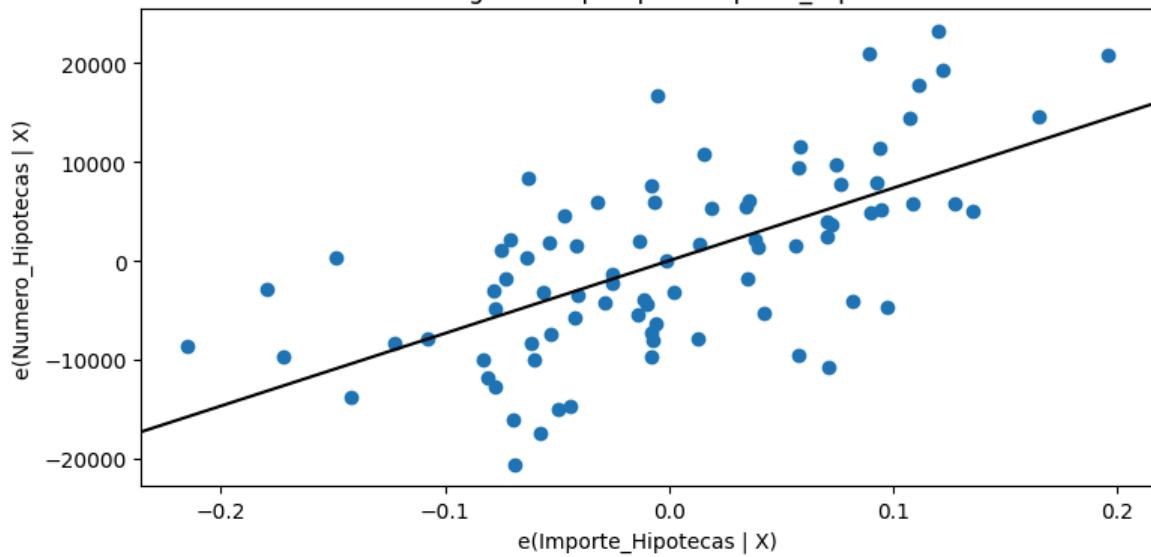
if len(independent_vars_temp2) == 1:
    axes = [axes]

for ax, var in zip(axes, independent_vars_temp2):
    # Generamos partial regression plot
    plot_partregress(endog="Numero_Hipotecas",
                      exog_i=var,
                      exog_others=[v for v in independent_vars_temp2 if v != var],
                      data=df_temp2, ax=ax)
    ax.set_title(f'Partial regression plot para {var}')

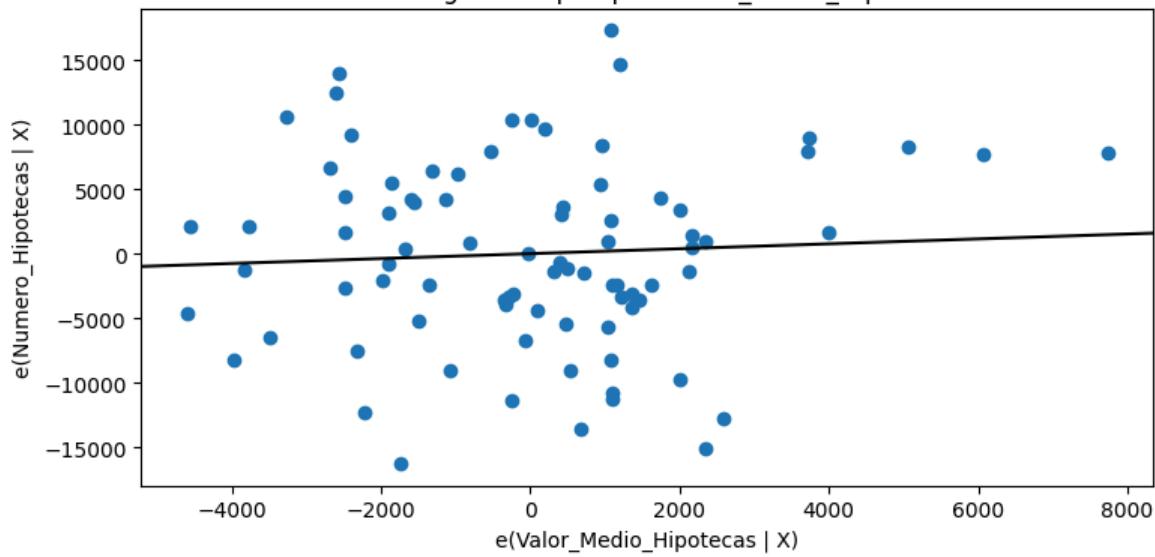
    # Quitar las etiquetas de puntos (textos)
    for txt in ax.texts:
        txt.set_visible(False)

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

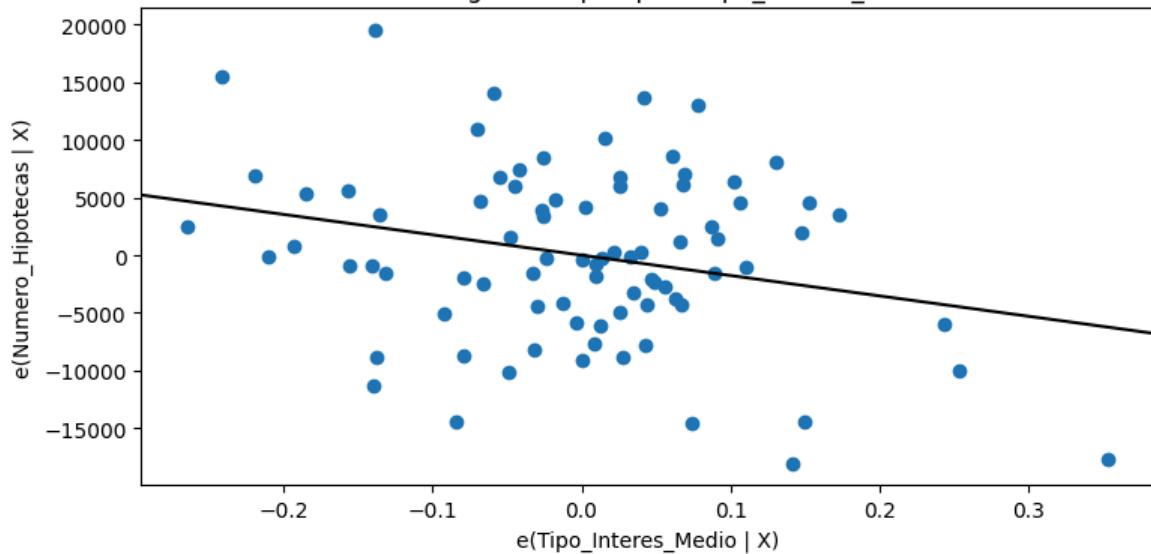
Partial regression plot para Importe_Hipotecas



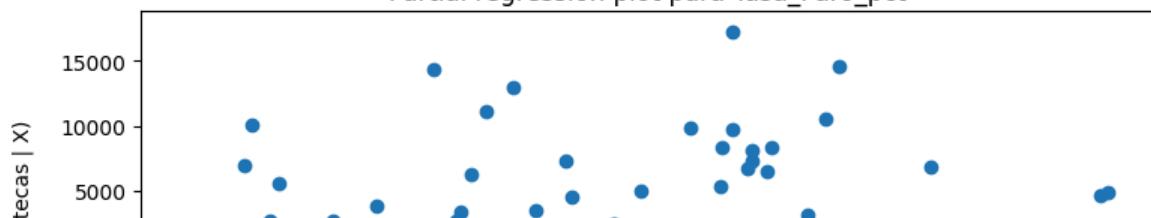
Partial regression plot para Valor_Medio_Hipotecas

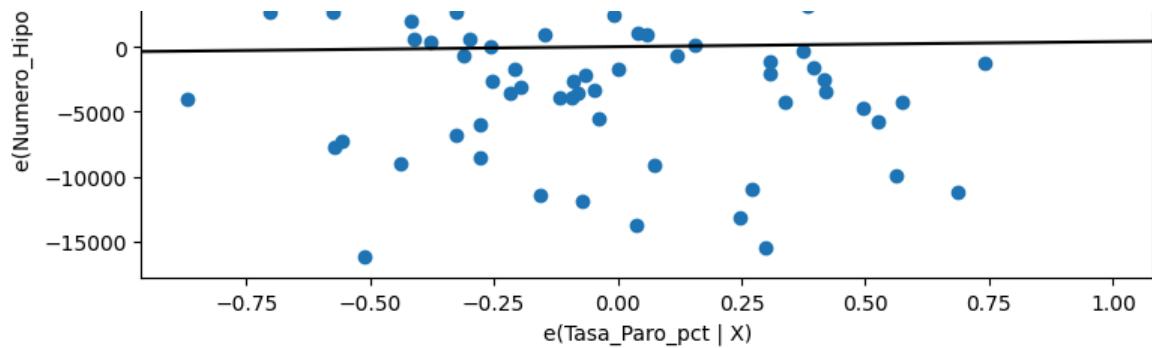


Partial regression plot para Tipo_Interes_Medio

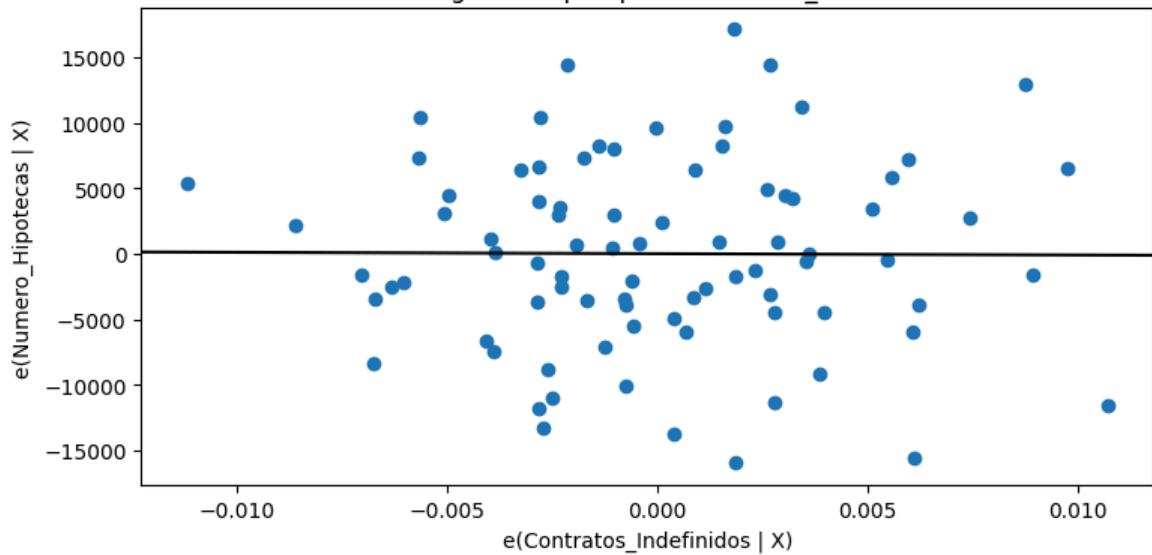


Partial regression plot para Tasa_Paro_pct

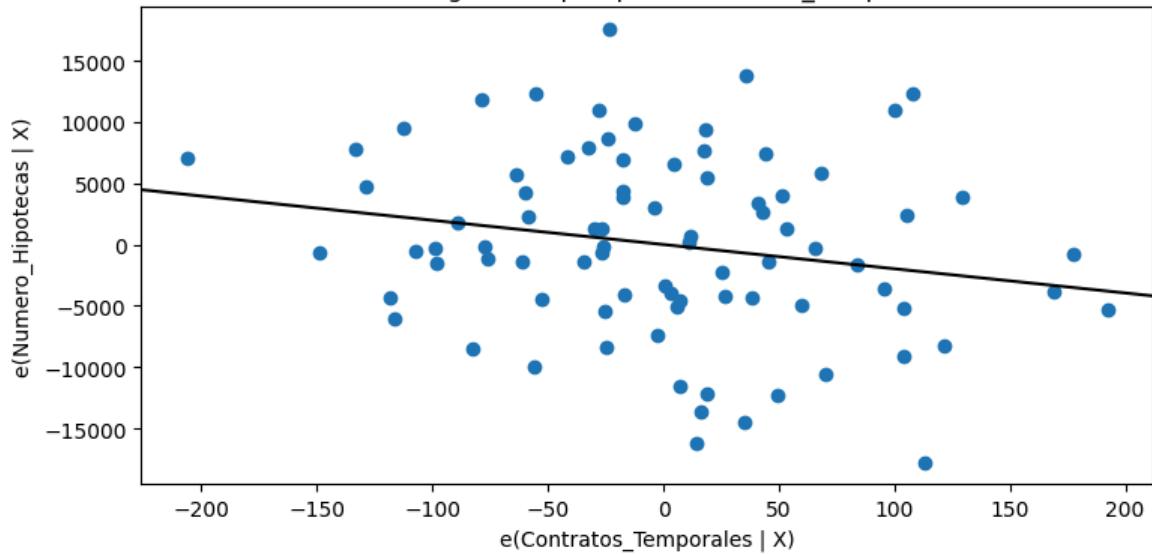




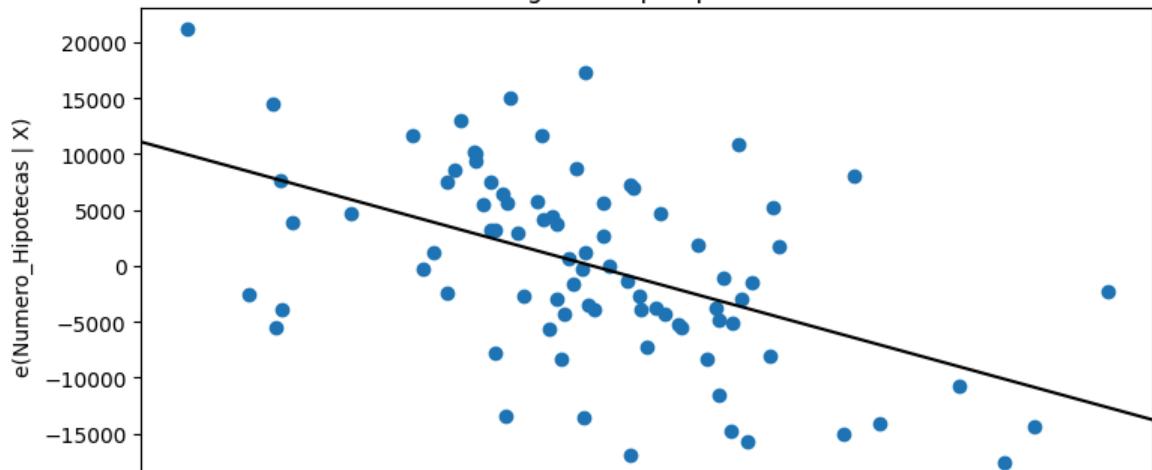
Partial regression plot para Contratos_Indefinidos

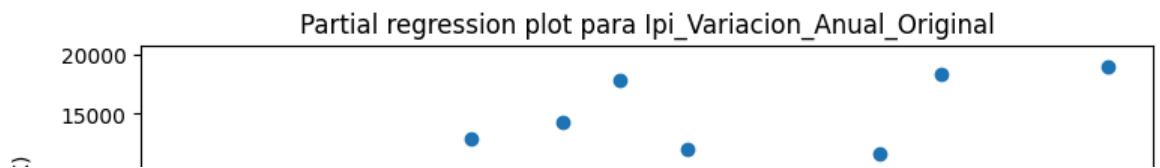
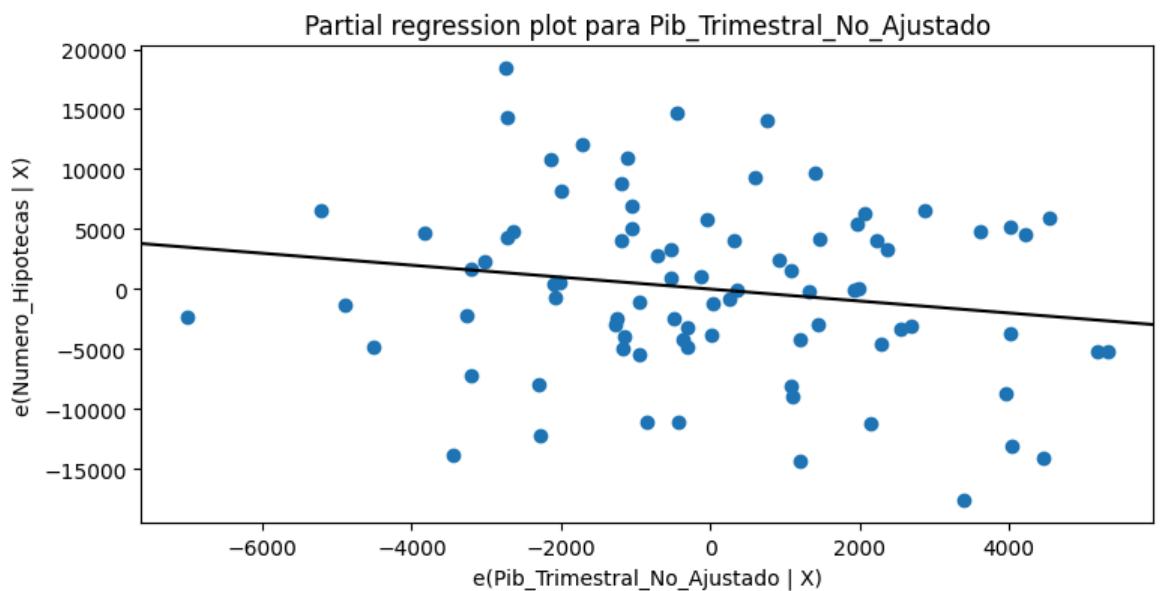
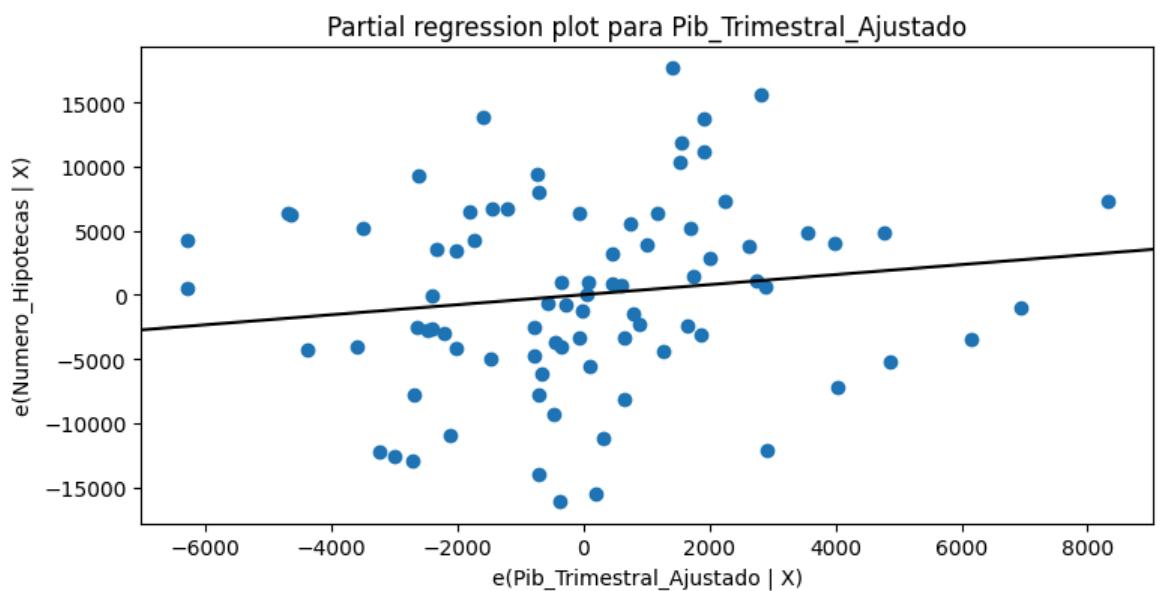
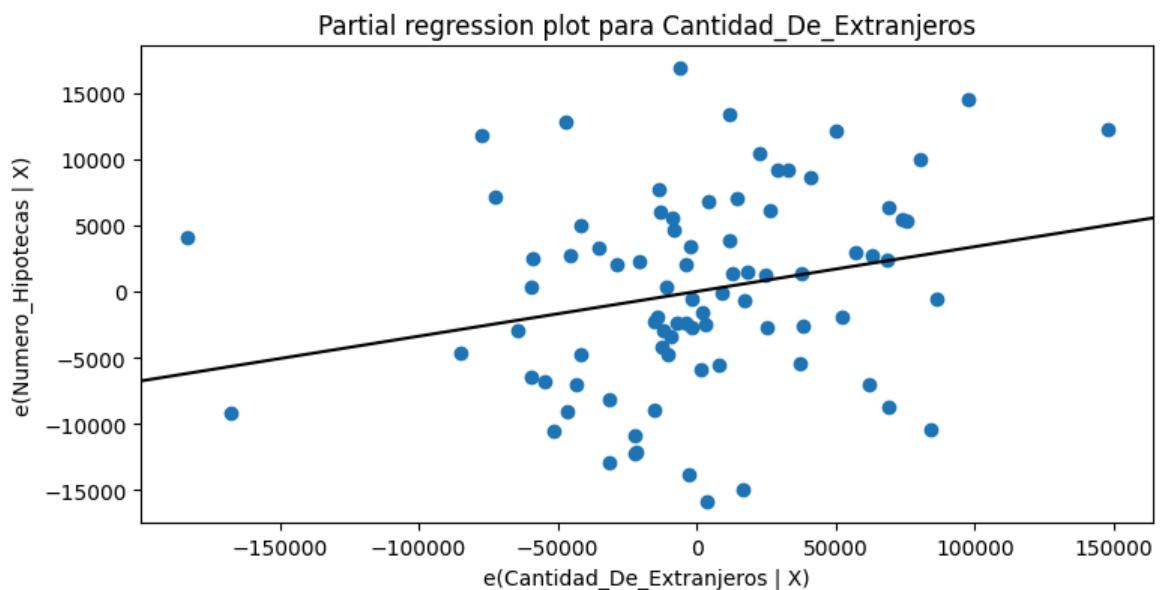
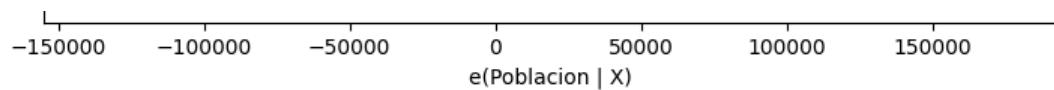


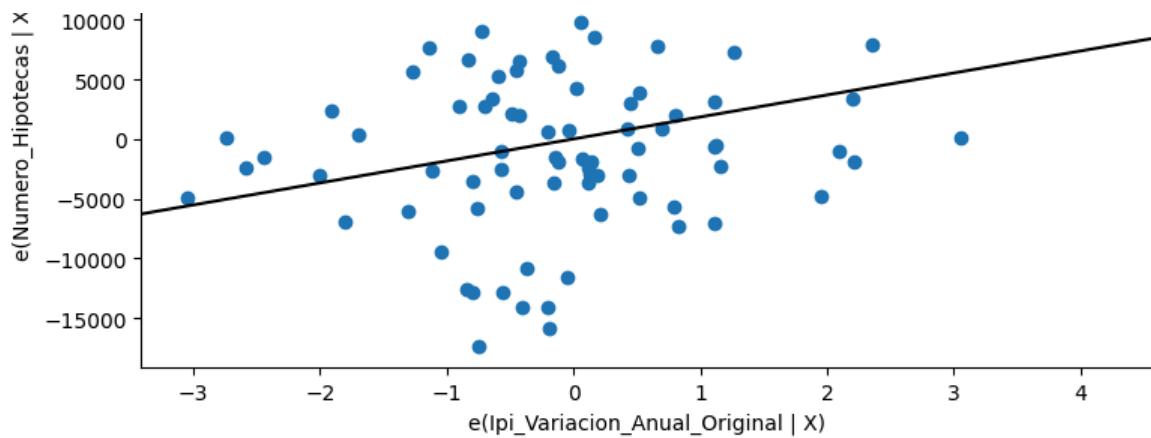
Partial regression plot para Contratos_Temporales



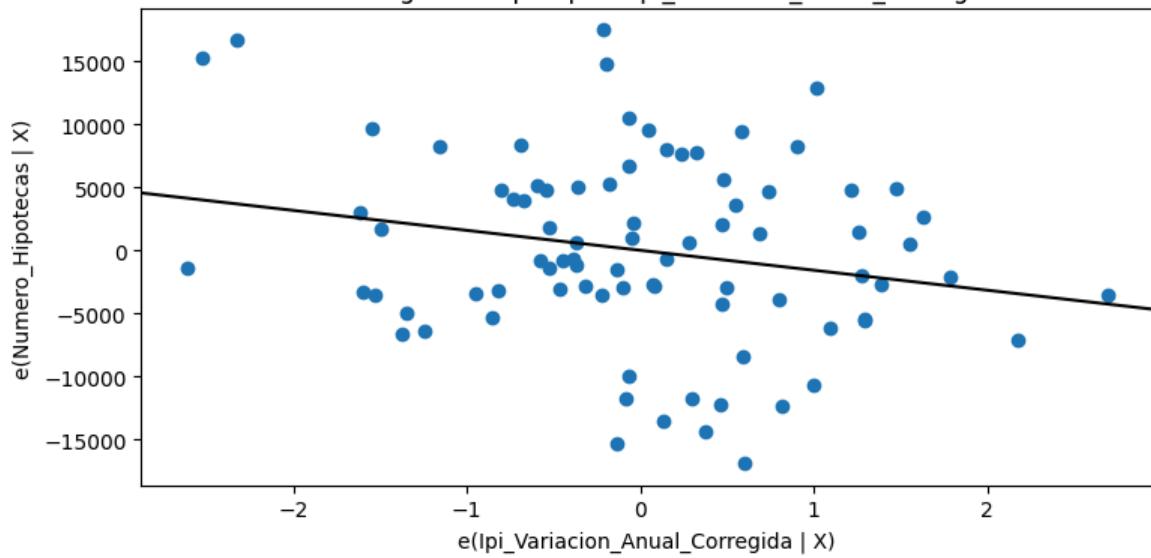
Partial regression plot para Poblacion



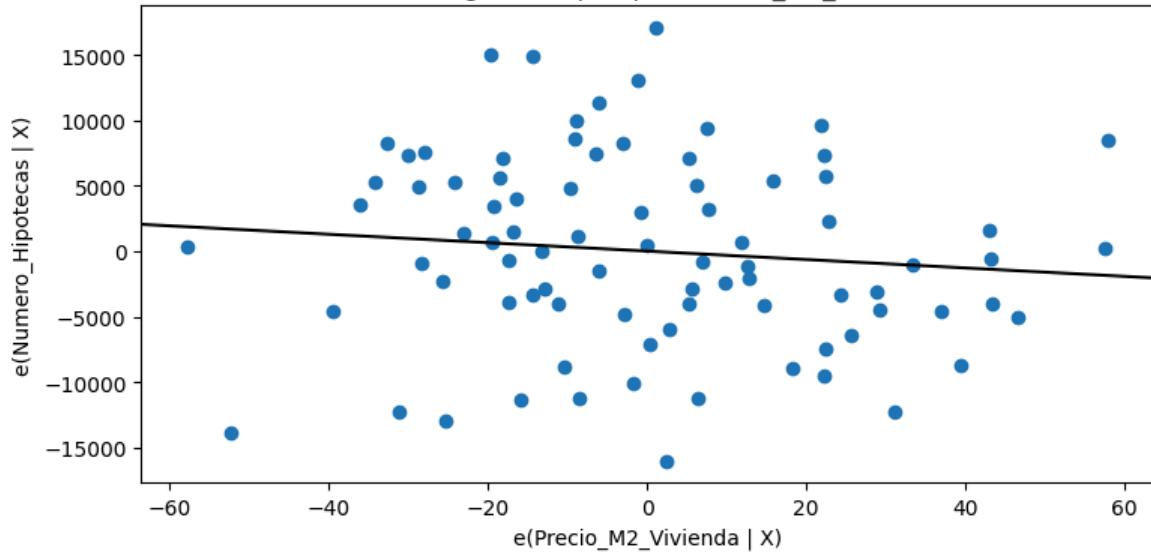




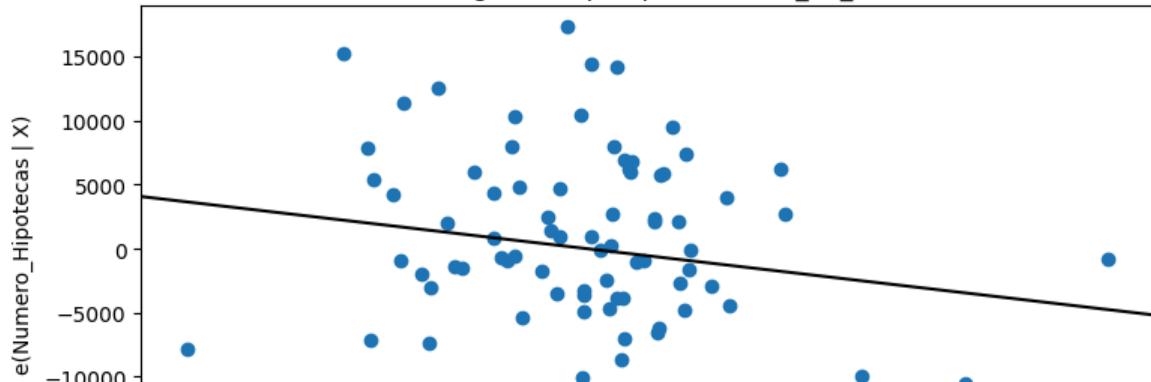
Partial regression plot para $IPI_{Variacion\ Annual\ Original} | X$

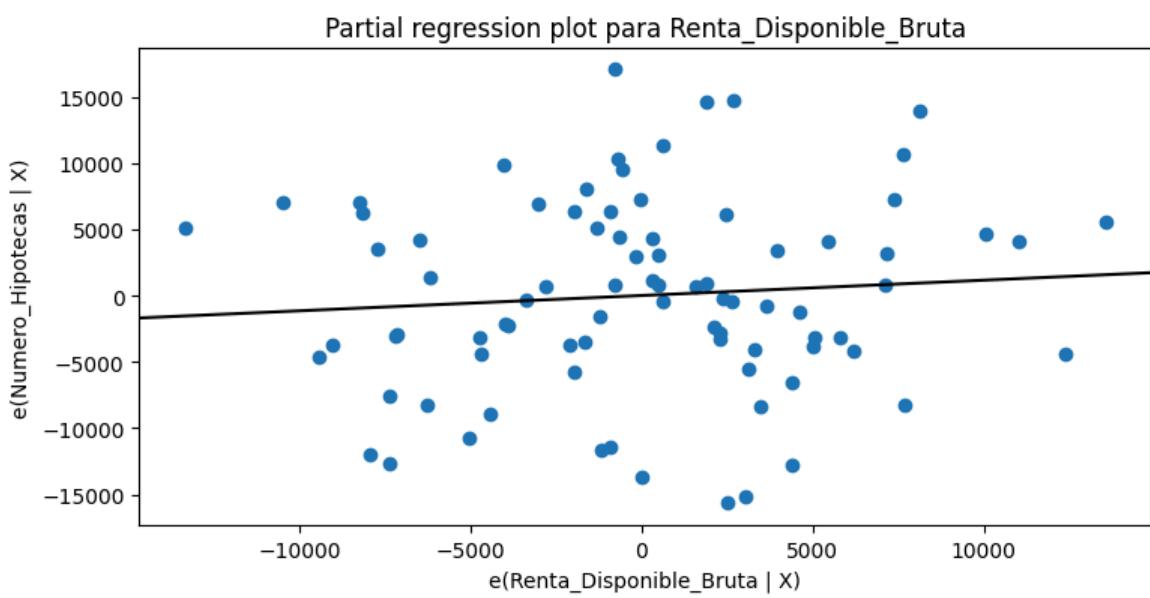
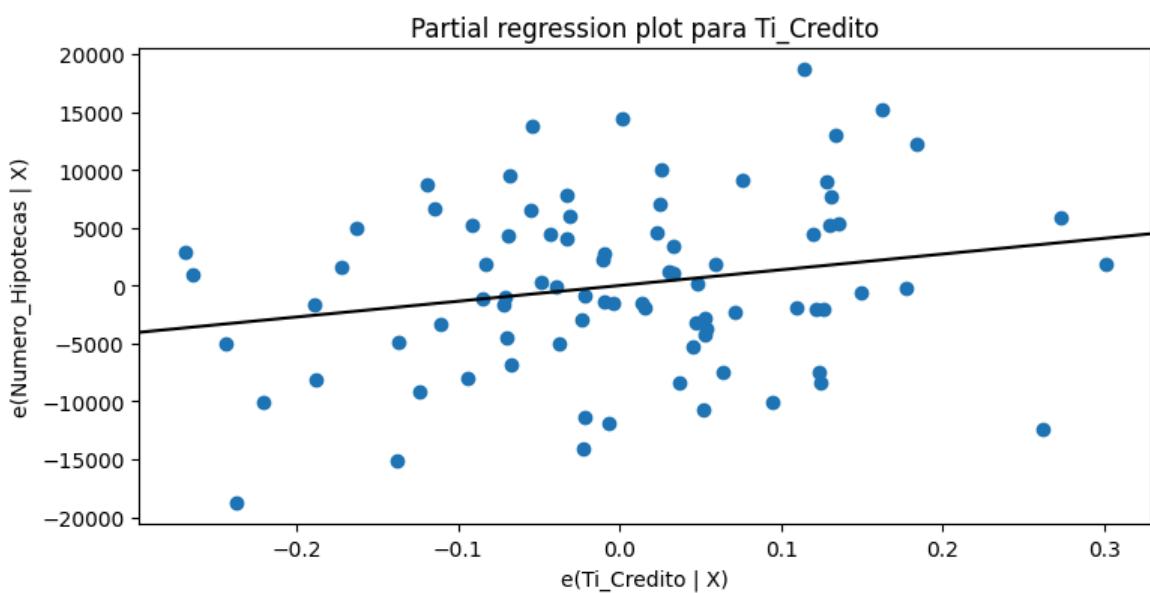
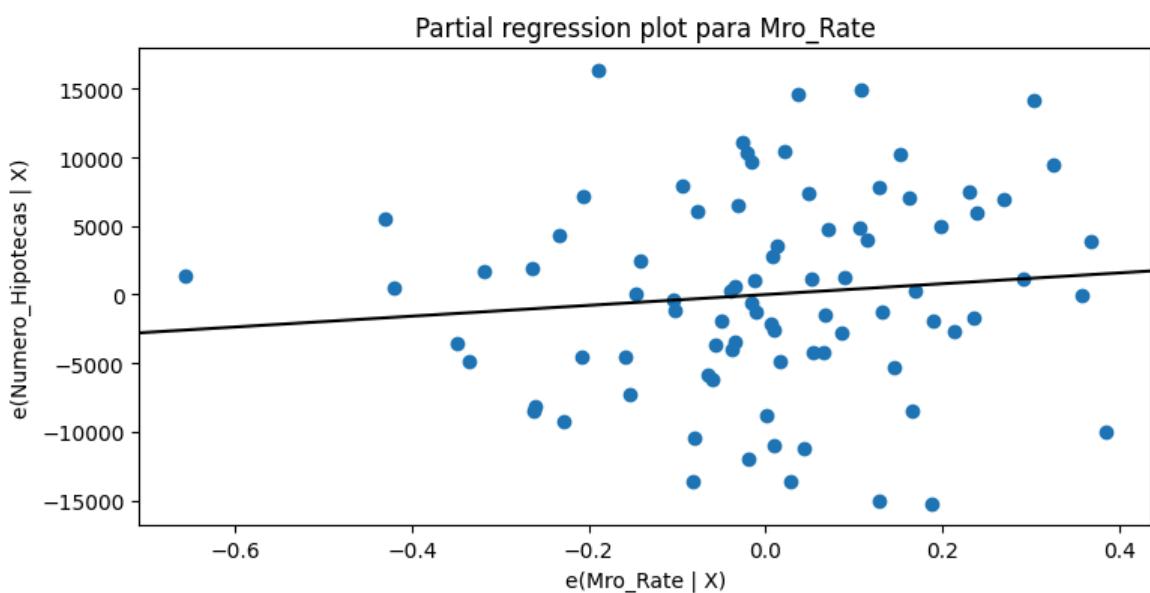
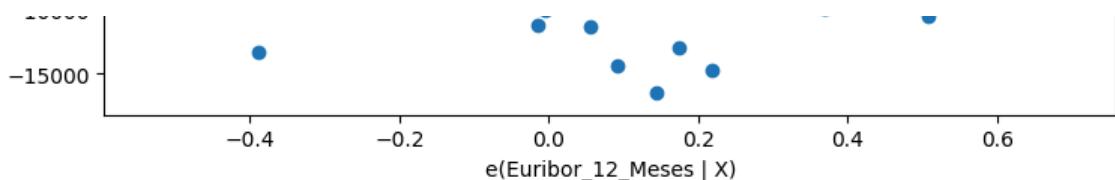


Partial regression plot para $IPI_{Variacion\ Annual\ Corregida} | X$

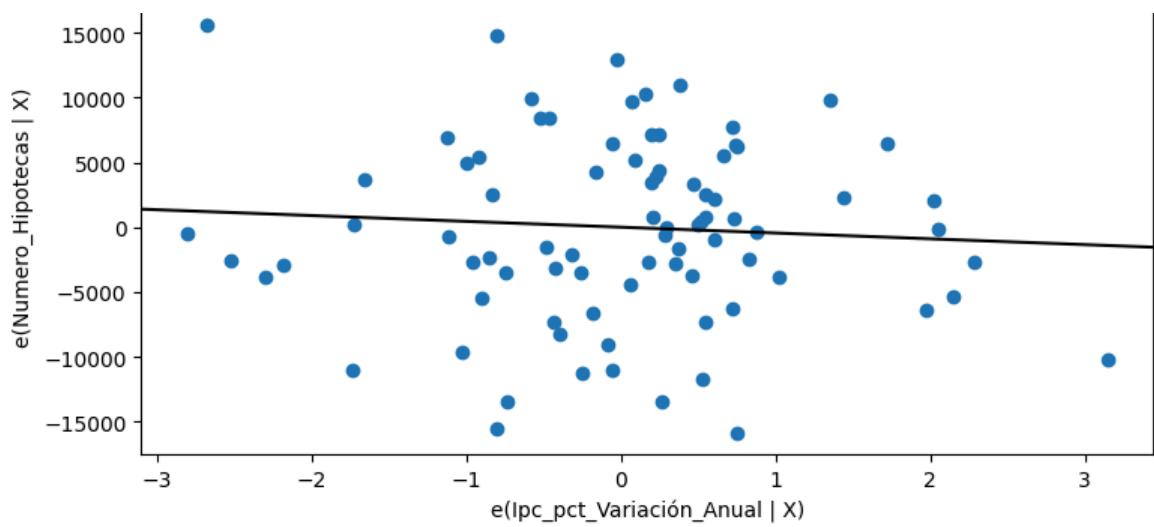


Partial regression plot para $Precio\ M2\ Vivienda | X$

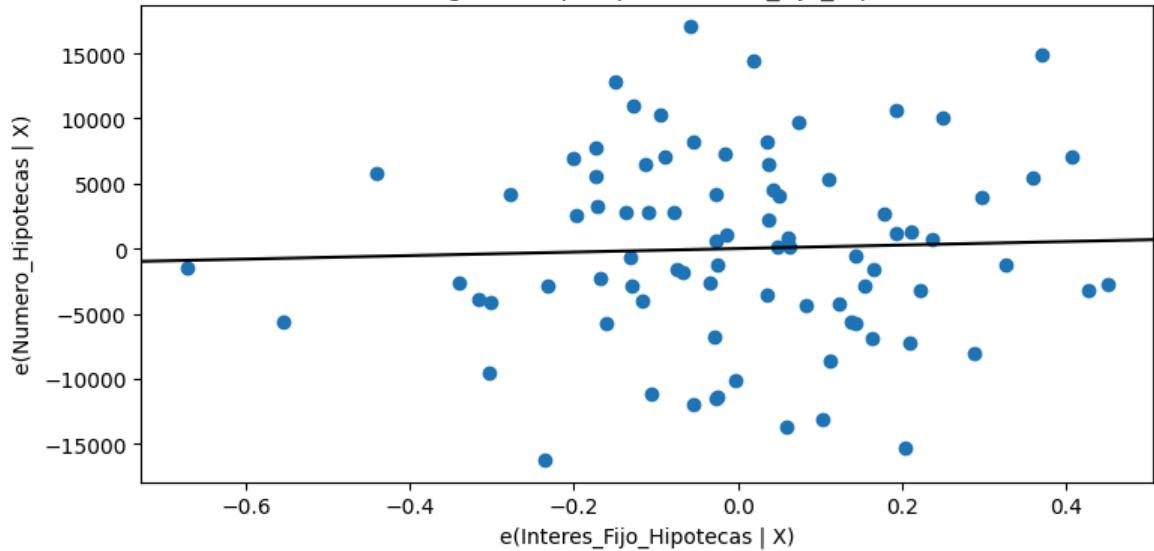




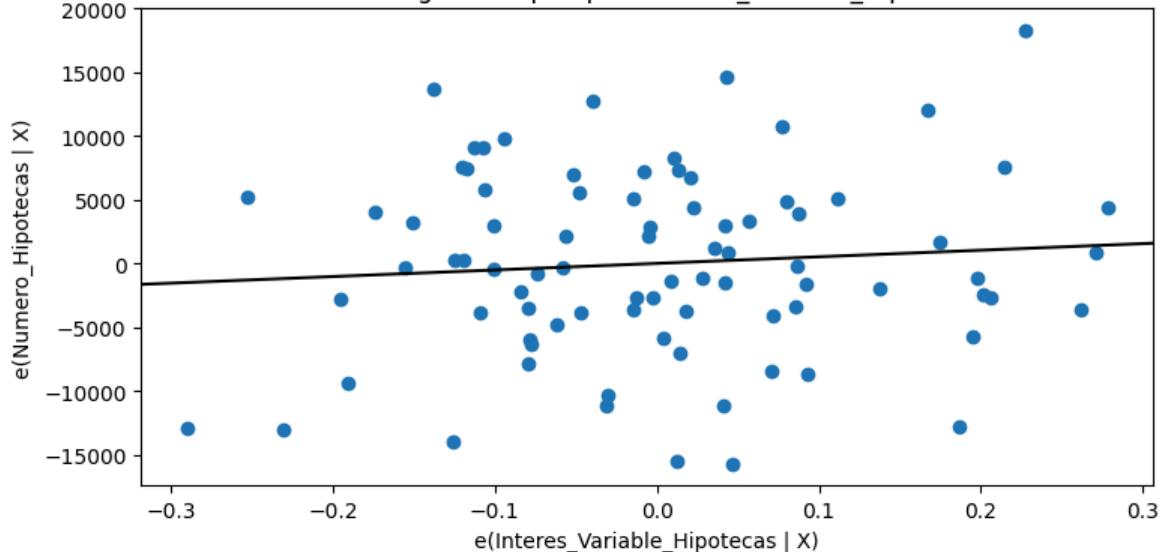
Partial regression plot para Ipc_pct_Variación_Anual



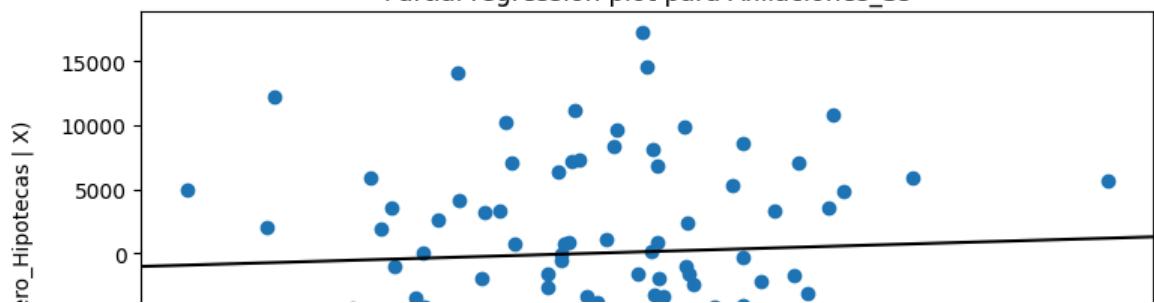
Partial regression plot para $e(\text{Numero_Hipotecas} | X)$



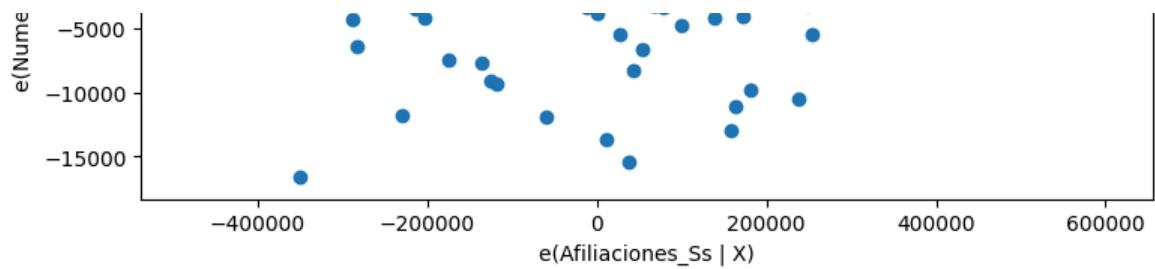
Partial regression plot para $e(\text{Interes_Fijo_Hipotecas} | X)$



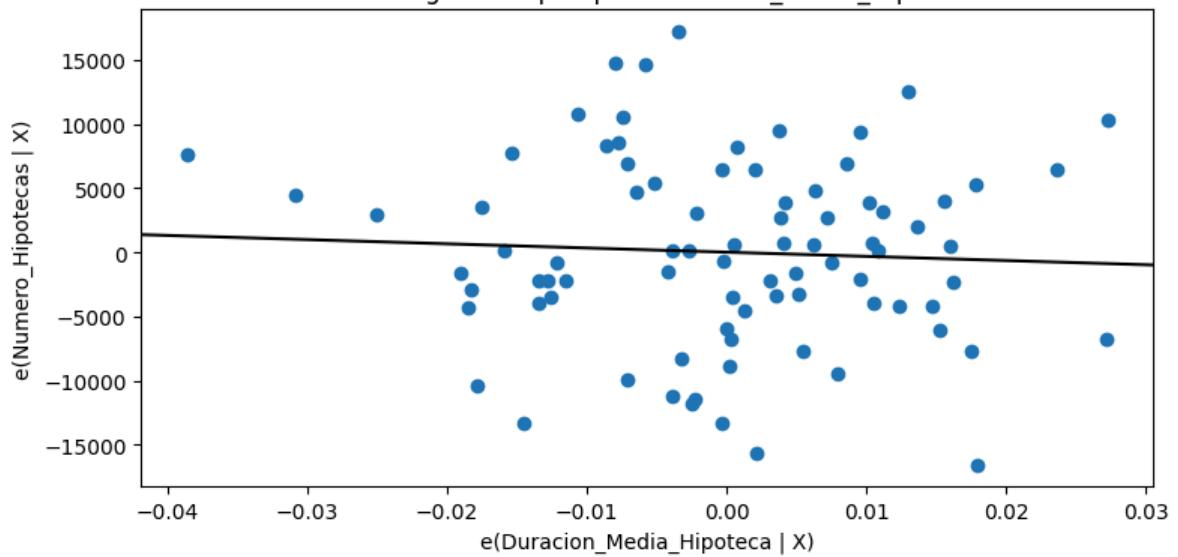
Partial regression plot para $e(\text{Interes_Variable_Hipotecas} | X)$



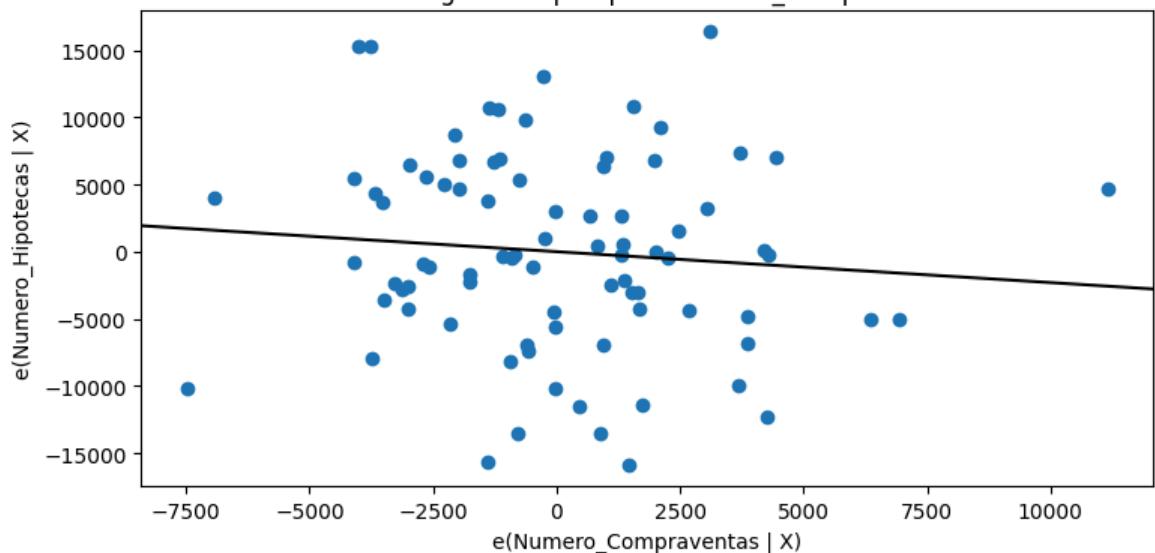
Partial regression plot para $e(\text{Afiliaciones_Ss} | X)$



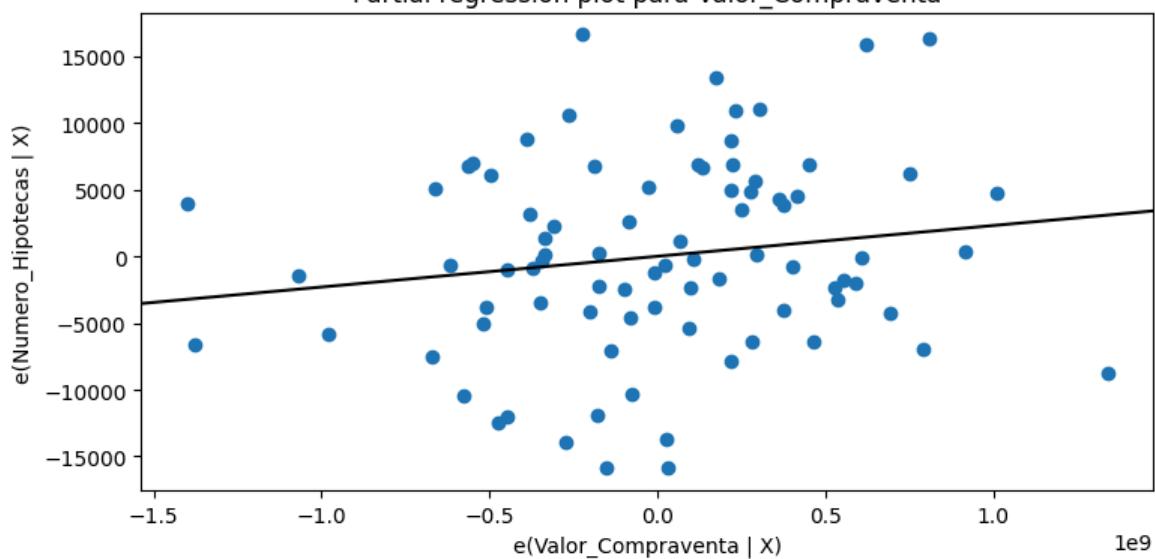
Partial regression plot para Duracion_Media_Hipoteca

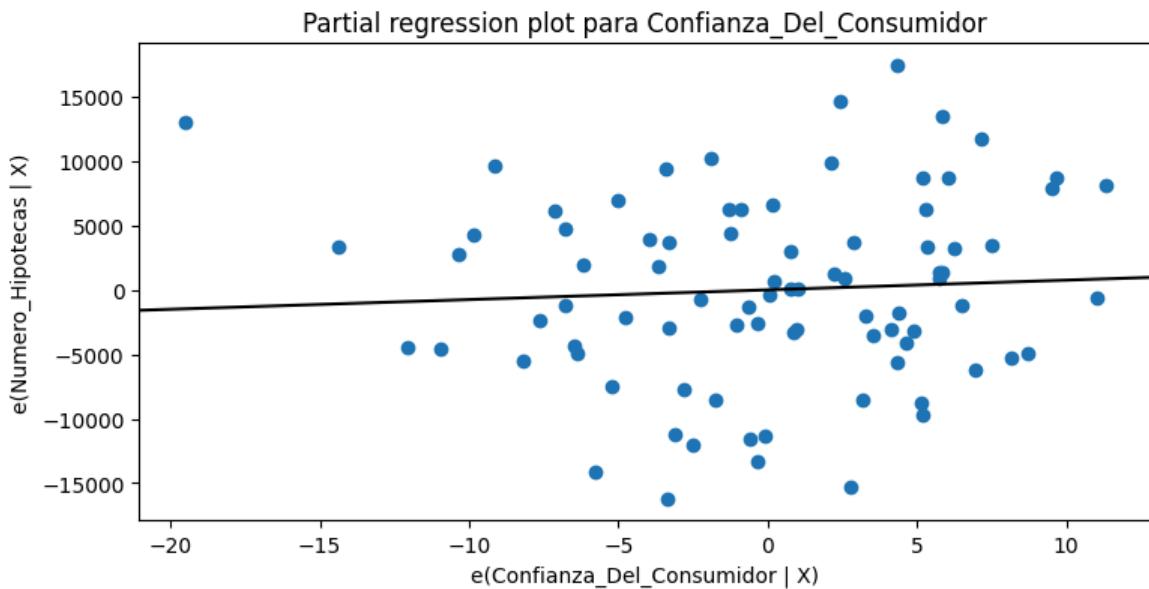


Partial regression plot para Numero_Compraventas



Partial regression plot para Valor_Compraventa





B) No multicolinealidad perfecta (VIF)

`df_no_corr`

```
In [13]: print("\n--- Supuesto (b): No multicolinealidad perfecta (VIF) ---")
vif = pd.DataFrame()
vif["Variable"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif)

# Excluye la constante al comprobar
max_vif = vif.loc[vif["Variable"] != "const", "VIF"].max()

if max_vif > 10:
    print("⚠️ Puede haber problemas de multicolinealidad (VIF > 10).")
else:
    print("✅ No se detecta multicolinealidad preocupante (VIF < 10).")

--- Supuesto (b): No multicolinealidad perfecta (VIF) ---
      Variable      VIF
0      const  29824.528622
1  Importe_Hipotecas  8.013996
2  Numero_Compraventas  5.201562
3  Precio_M2_Vivienda  4.524857
4  Contratos_Temporales  4.068406
5  Ipc_%_Variación_Annual  1.637125
6  Duracion_Media_Hipoteca  6.146076
7      Tasa_Paro_(%)  6.320993
✅ No se detecta multicolinealidad preocupante (VIF < 10).
```

`iqr_winsorized_df`

```
In [14]: print("\n--- Supuesto (b): No multicolinealidad perfecta (VIF) ---")
vif = pd.DataFrame()
vif["Variable"] = X2.columns
vif["VIF"] = [variance_inflation_factor(X2.values, i) for i in range(X2.shape[1])]
print(vif)

if vif['VIF'].max() > 10:
    print("⚠️ Puede haber problemas de multicolinealidad (VIF > 10).")
else:
    print("✅ No se detecta multicolinealidad preocupante (VIF < 10).")
```

```
--- Supuesto (b): No multicolinealidad perfecta (VIF) ---
      Variable          VIF
0           const  3.937241e+06
1      Importe_Hipotecas  6.949783e+01
2    Valor_Medio_Hipotecas  4.968165e+01
3      Tipo_Interes_Medio  6.324364e+01
4      Tasa_Paro_(%)  1.876463e+02
5  Contratos_Indefinidos  4.030348e+02
6  Contratos_Temporales  8.321645e+01
7        Poblacion  3.782105e+02
8  Cantidad_De_Extranjeros  2.343229e+02
9  Pib_Trimestral_Ajustado  1.901425e+02
10  Pib_Trimestral_No_Ajustado  2.035008e+02
11  Ipi_Variacion_Anual_Original  9.601749e+00
12  Ipi_Variacion_Anual_Corregida  1.290828e+01
13      Precio_M2_Vivienda  6.786734e+01
14      Euribor_12_Meses  7.886384e+01
15      Mro_Rate  6.418439e+01
16      Ti_Credito  8.177181e+01
17  Renta_Disponible_Bruta  2.436555e+01
18      Ipc_%_Variación_Anual  3.204117e+00
19      Interes_Fijo_Hipotecas  3.659195e+01
20  Interes_Variable_Hipotecas  5.211976e+01
21      Afiliaciones_Ss  3.609592e+02
22  Duracion_Media_Hipoteca  1.501733e+01
23      Numero_Compraventas  2.781176e+02
24      Valor_Compraventa  3.053069e+02
25  Confianza_Del_Consumidor  8.367266e+00
```

⚠ Puede haber problemas de multicolinealidad (VIF > 10).

C) Esperanza cero del error

`df_no_corr`

```
In [15]: residuals = model.resid
mean_resid = np.mean(residuals)

print("\n--- Supuesto (c): Esperanza cero del error ---")
print(f"Media de los residuos: {mean_resid:.5f}")
if abs(mean_resid) < 0.01:
    print("✅ Media de residuos próxima a cero.")
else:
    print("⚠ Media de residuos algo alejada de cero.")
```

--- Supuesto (c): Esperanza cero del error ---

Media de los residuos: 0.00000

✅ Media de residuos próxima a cero.

`iqr_winsorized_df`

```
In [16]: residuals2 = model2.resid
mean_resid2 = np.mean(residuals2)

print("\n--- Supuesto (c): Esperanza cero del error ---")
print(f"Media de los residuos: {mean_resid2:.5f}")
if abs(mean_resid2) < 0.01:
    print("✅ Media de residuos próxima a cero.")
else:
    print("⚠ Media de residuos algo alejada de cero.")
```

--- Supuesto (c): Esperanza cero del error ---

Media de los residuos: -0.08119

⚠ Media de residuos algo alejada de cero.

D) Homocedasticidad

`df_no_corr`

```
In [17]: print("\n--- Supuesto (d): Homocedasticidad (Breusch-Pagan test) ---")
bp_test = het_breuscpagan(residuals, X)
labels = ['LM stat', 'LM p-val', 'F stat', 'F p-val']
for l, r in zip(labels, bp_test):
    print(f"{l}: {r:.4f}")

if bp_test[1] < 0.05:
    print("⚠ Se rechaza homocedasticidad: posible heterocedasticidad.")
else:
    print("✅ No se rechaza homocedasticidad.")
```

```
--- Supuesto (d): Homocedasticidad (Breusch-Pagan test) ---
LM stat: 8.7574
LM p-val: 0.2705
F stat: 1.2636
F p-val: 0.2797
✓ No se rechaza homocedasticidad.
```

iqr_winsorized_df

```
In [18]: print("\n--- Supuesto (d): Homocedasticidad (Breusch-Pagan test) ---")
bp_test2 = het_breuschpagan(residuals2, X2)
labels2 = ['LM stat', 'LM p-val', 'F stat', 'F p-val']
for l, r in zip(labels2, bp_test2):
    print(f'{l}: {r:.4f}')

if bp_test2[1] < 0.05:
    print("⚠ Se rechaza homocedasticidad: posible heterocedasticidad.")
else:
    print("✓ No se rechaza homocedasticidad.")

--- Supuesto (d): Homocedasticidad (Breusch-Pagan test) ---
LM stat: 31.4902
LM p-val: 0.1733
F stat: 1.3913
F p-val: 0.1505
✓ No se rechaza homocedasticidad.
```

E) No autocorrelación de los errores

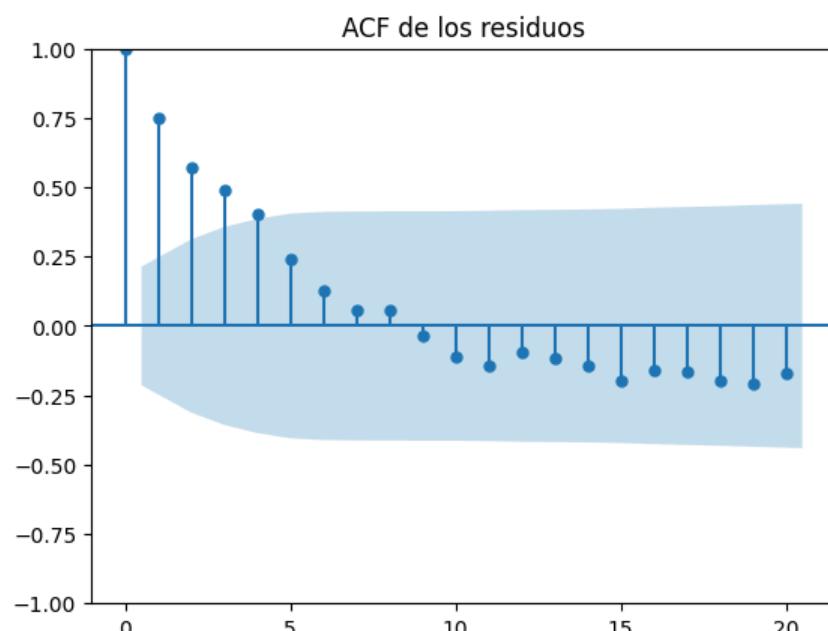
df_no_corr

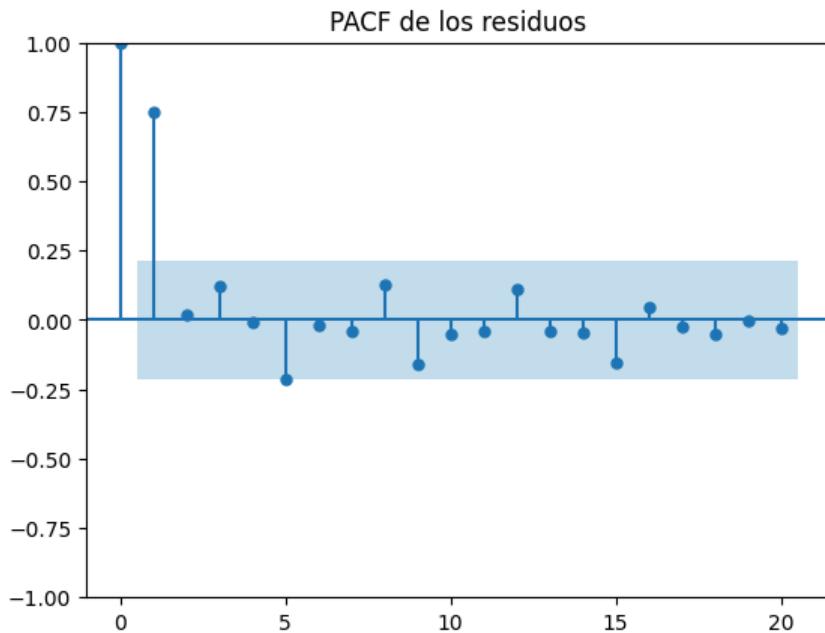
```
In [19]: print("\n--- Supuesto (e): No autocorrelación (Durbin-Watson) ---")
dw = durbin_watson(residuals)
print(f"Durbin-Watson: {dw:.4f}")
if 1.5 < dw < 2.5:
    print("✓ No hay indicios claros de autocorrelación.")
else:
    print("⚠ Posibles problemas de autocorrelación.")

plot_acf(residuals)
plt.title("ACF de los residuos")
plt.show()

plot_pacf(residuals)
plt.title("PACF de los residuos")
plt.show()

--- Supuesto (e): No autocorrelación (Durbin-Watson) ---
Durbin-Watson: 0.4591
⚠ Posibles problemas de autocorrelación.
```





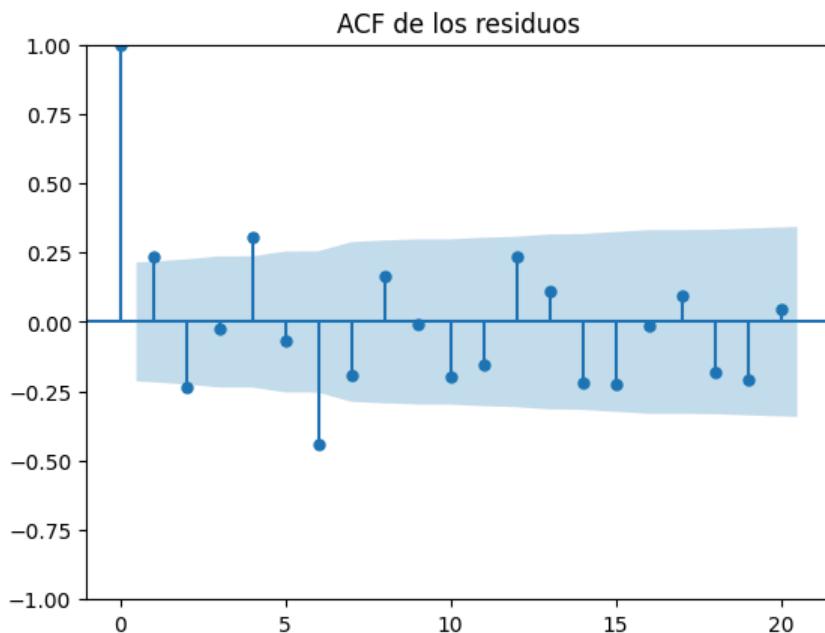
`iqr_winsorized_df`

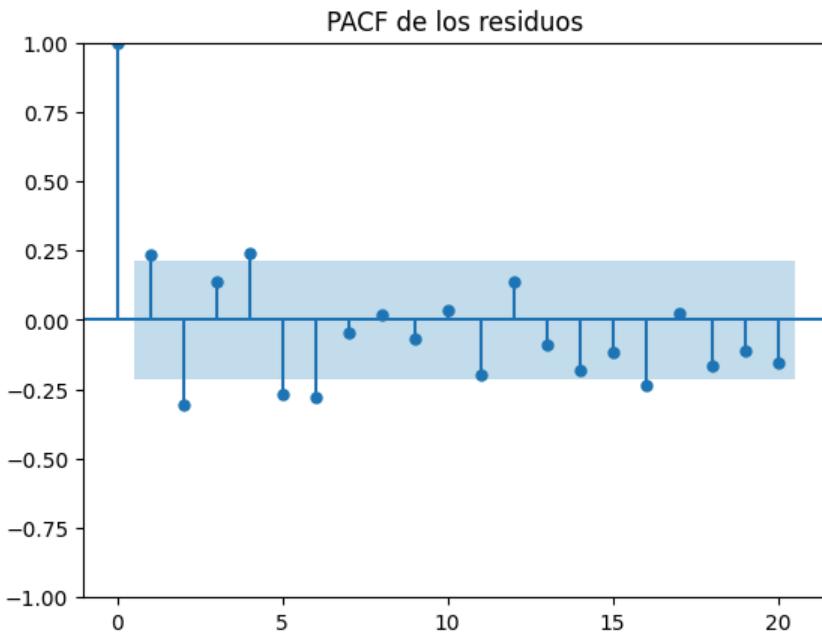
```
In [20]: print("\n--- Supuesto (e): No autocorrelación (Durbin-Watson) ---")
dw2 = durbin_watson(residuals2)
print(f"Durbin-Watson: {dw2:.4f}")
if 1.5 < dw2 < 2.5:
    print("✅ No hay indicios claros de autocorrelación.")
else:
    print("⚠️ Posibles problemas de autocorrelación.")

plot_acf(residuals2)
plt.title("ACF de los residuos")
plt.show()

plot_pacf(residuals2)
plt.title("PACF de los residuos")
plt.show()
```

--- Supuesto (e): No autocorrelación (Durbin-Watson) ---
Durbin-Watson: 1.5188
✅ No hay indicios claros de autocorrelación.





F) Normalidad de los errores

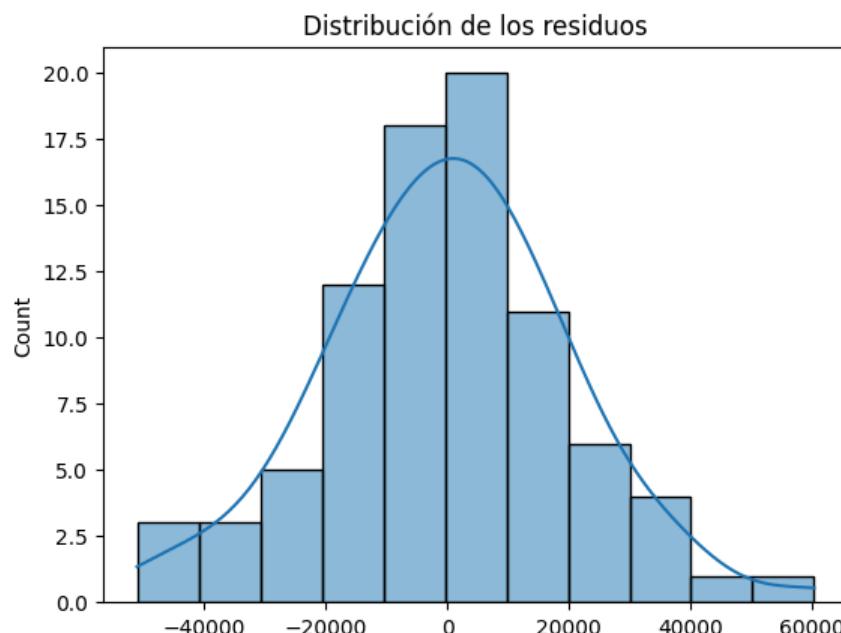
`df_no_corr`

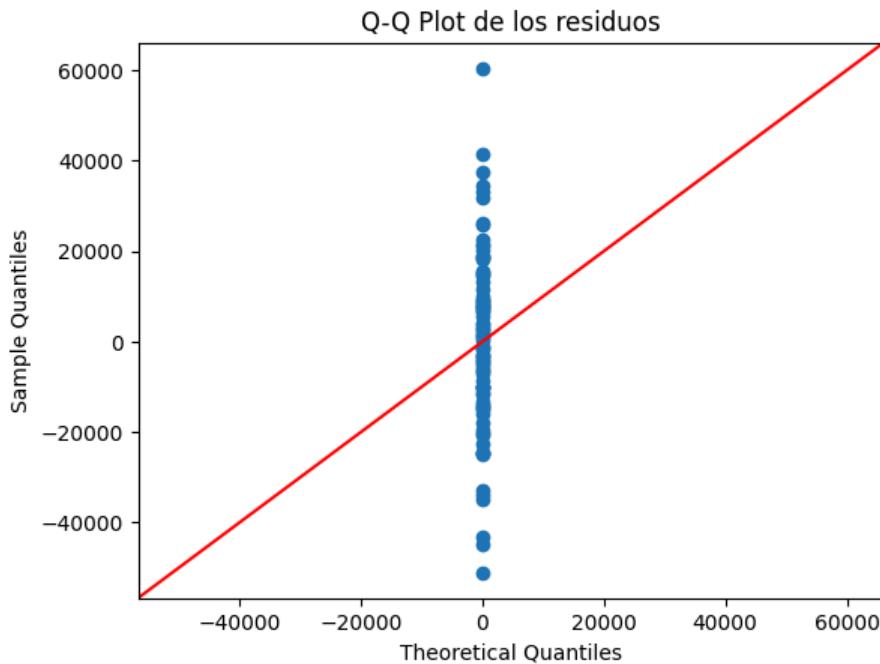
```
In [21]: print("\n--- Supuesto (f): Normalidad de los residuos (Shapiro-Wilk) ---")
shap = shapiro(residuals)
print(f"Estadístico: {shap[0]:.4f}, p-valor: {shap[1]:.4f}")
if shap[1] < 0.05:
    print("⚠ Se rechaza normalidad de los residuos.")
else:
    print("✅ No se rechaza normalidad de los residuos.")

sns.histplot(residuals, kde=True)
plt.title("Distribución de los residuos")
plt.show()

sm.qqplot(residuals, line='45')
plt.title("Q-Q Plot de los residuos")
plt.show()
```

--- Supuesto (f): Normalidad de los residuos (Shapiro-Wilk) ---
 Estadístico: 0.9935, p-valor: 0.9545
 ✅ No se rechaza normalidad de los residuos.





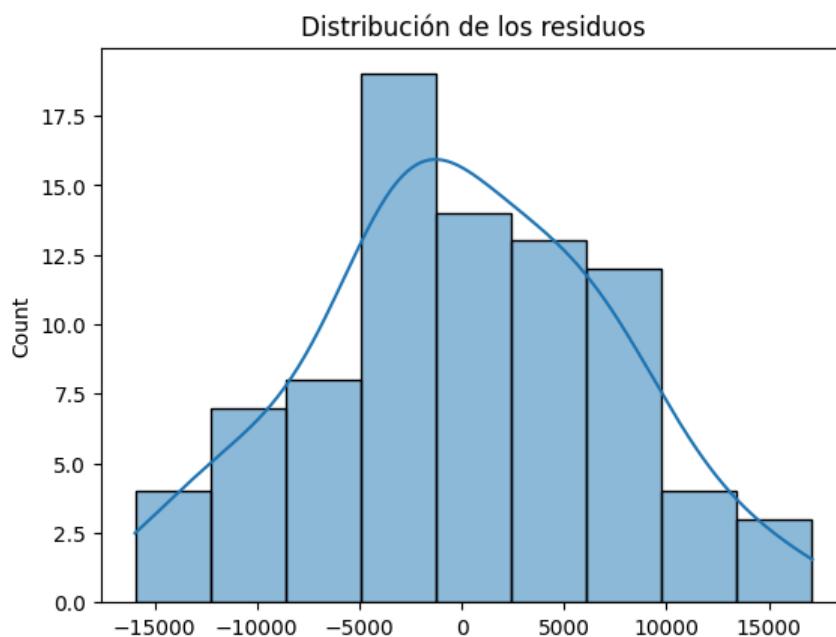
`iqr_winsorized_df`

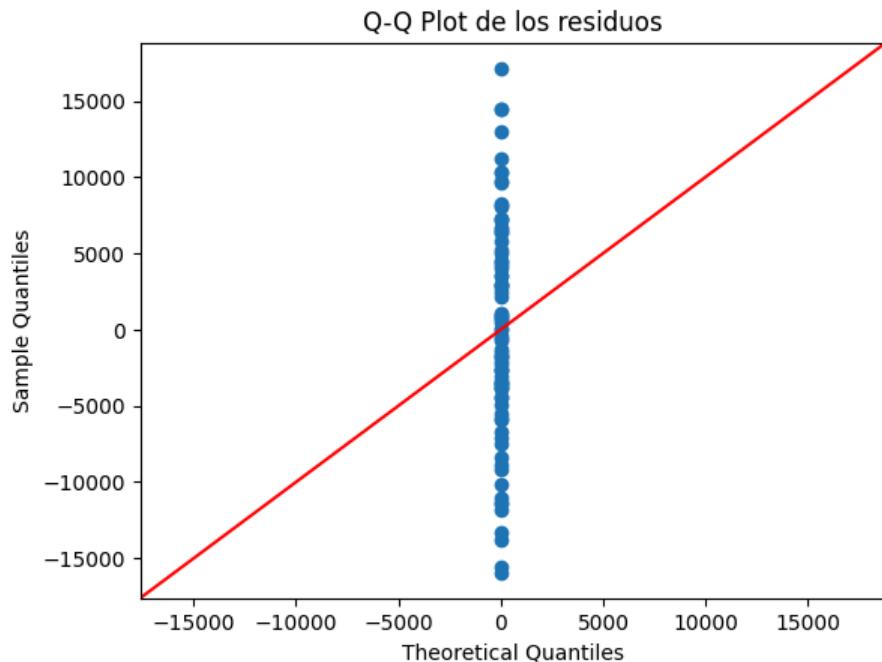
```
In [22]: print("\n--- Supuesto (f): Normalidad de los residuos (Shapiro-Wilk) ---")
shapiro2 = shapiro(residuals2)
print(f"Estadístico: {shapiro2[0]:.4f}, p-valor: {shapiro2[1]:.4f}")
if shapiro2[1] < 0.05:
    print("⚠ Se rechaza normalidad de los residuos.")
else:
    print("✅ No se rechaza normalidad de los residuos.")

sns.histplot(residuals2, kde=True)
plt.title("Distribución de los residuos")
plt.show()

sm.qqplot(residuals2, line='45')
plt.title("Q-Q Plot de los residuos")
plt.show()
```

--- Supuesto (f): Normalidad de los residuos (Shapiro-Wilk) ---
 Estadístico: 0.9926, p-valor: 0.9212
 ✅ No se rechaza normalidad de los residuos.





E) Resultados y limitaciones del modelo

El conjunto de datos para el cual son válidos un mayor número de supuestos es `df_no_corr`, obtenido tras la eliminación de las variables con mayor correlación (entre sí y con la variable objetivo) en el anterior notebook.

Este dataset **cumple** con:

- No multicolinealidad preocupante.
- Homocedasticidad.
- Normalidad de los errores.
- Esperanza cero del error.

Sin embargo, **no cumple** con:

- Linealidad de los parámetros: la relación entre la variable dependiente (`Numero_Hipotecas`) y las variables independientes no está bien capturada mediante una relación lineal. Por tanto, los coeficientes estimados no representan correctamente el efecto marginal de cada variable. Es decir, un coeficiente β no indica el cambio esperado en Y ante un cambio unitario en X , porque la relación no es lineal.
- No autocorrelación de los errores: esto indica que los errores no son independientes y, por tanto, que las predicciones pueden ser menos fiables, sobre todo para series temporales, como es nuestro caso.

Tras observar los gráficos partial regression plots del apartado **A) Linealidad de los parámetros** podemos decir que las variables con patrones que sugieren una posible no linealidad son: `Numero_Compraventas`, `IPC_%_Variacion_Anual`, `Duracion_Media_Hipoteca` y `Contratos_Temporales`, ya que muestran una nubes de puntos con elevada dispersión y patrones que no son claros. Por tanto, vamos a aplicar **distintas transformaciones (logarítmicas, cuadrados...)** a dichas variables para ver si así conseguimos mejorar la linealidad de los parámetros.

```
In [23]: df_no_corr2 = df_no_corr.copy()
df_no_corr2['log_Contratos_Temporales'] = np.log1p(df_no_corr2['Contratos_Temporales'] - df_no_corr2['Contratos_Temporales'].min())
df_no_corr2['log_Numero_Compraventas'] = np.log1p(df_no_corr2['Numero_Compraventas'])
df_no_corr2['Ipc_%_Variación_Anual_cuadrado'] = df_no_corr2['Ipc_%_Variación_Anual'] ** 2
df_no_corr2['sqrt_Duracion_Media_Hipoteca'] = np.sqrt(df_no_corr2['Duracion_Media_Hipoteca'] - df_no_corr2['Duracion_Media_Hipoteca'].min())
df_no_corr2 = df_no_corr2.drop(['Contratos_Temporales', 'Numero_Compraventas', 'Ipc_%_Variación_Anual', 'Duracion_Media_Hipoteca'], axis=1)
df_no_corr2.head()
```

Out[23]:

Fecha	Numero_Hipotecas	Importe_Hipotecas	Precio_M2_Vivienda	Tasa_Paro_(%)	log_Contratos_Temporales	log_Número_C...
2003-03-31	263600	23.908609	1230.3	11.99	7.216416	
2003-06-30	247071	23.876021	1309.6	11.28	7.341030	
2003-09-30	236977	23.900416	1344.9	11.30	7.404644	
2003-12-31	241791	23.926874	1380.3	11.37	7.425894	
2004-03-31	283170	24.125194	1456.2	11.50	7.385293	

In [24]:

```
df3 = df_no_corr2.copy()

# Ajuste del modelo
X3 = df3.drop(columns=['Numero_Hipotecas'])
y3 = df3['Numero_Hipotecas']
X3 = sm.add_constant(X3)

model3 = sm.OLS(y3, X3).fit()

# Coeficientes
print("--- Coeficientes del modelo ---")
print(model3.params)

--- Coeficientes del modelo ---
const              -4.848632e+06
Importe_Hipotecas  1.788336e+05
Precio_M2_Vivienda -1.347996e+02
Tasa_Paro_(%)      6.318497e+03
log_Contratos_Temporales -1.175551e+03
log_Número_Compraventas  2.921793e+04
Ipc_%_Variación_Annual_cuadrado -7.618683e+02
sqrt_Duracion_Media_Hipoteca  5.788712e+05
dtype: float64
```

In [25]:

```
reset = linear_reset(model3, power=2, use_f=True)
print("--- Ramsey RESET test ---")
print(f" F-stat: {reset.fvalue:.4f}, p-value: {reset.pvalue:.4f}")

if reset.pvalue < 0.05:
    print("⚠ Se rechaza H0: posible falta de linealidad o forma funcional incorrecta.")
else:
    print("✅ No se rechaza H0: no hay indicios de falta de linealidad o forma funcional incorrecta.")

--- Ramsey RESET test ---
F-stat: 527.3797, p-value: 0.0000
⚠ Se rechaza H0: posible falta de linealidad o forma funcional incorrecta.
```

In [26]:

```
df_temp3 = df_no_corr2.rename(
    columns=lambda x: x.replace(" ", "_").replace("(", "").replace(")", "").replace("%", "pct")
)

independent_vars_temp3 = [col for col in df_temp3.columns if col != "Numero_Hipotecas"]

fig, axes = plt.subplots(len(independent_vars_temp3), 1, figsize=(8, 4 * len(independent_vars_temp3)))

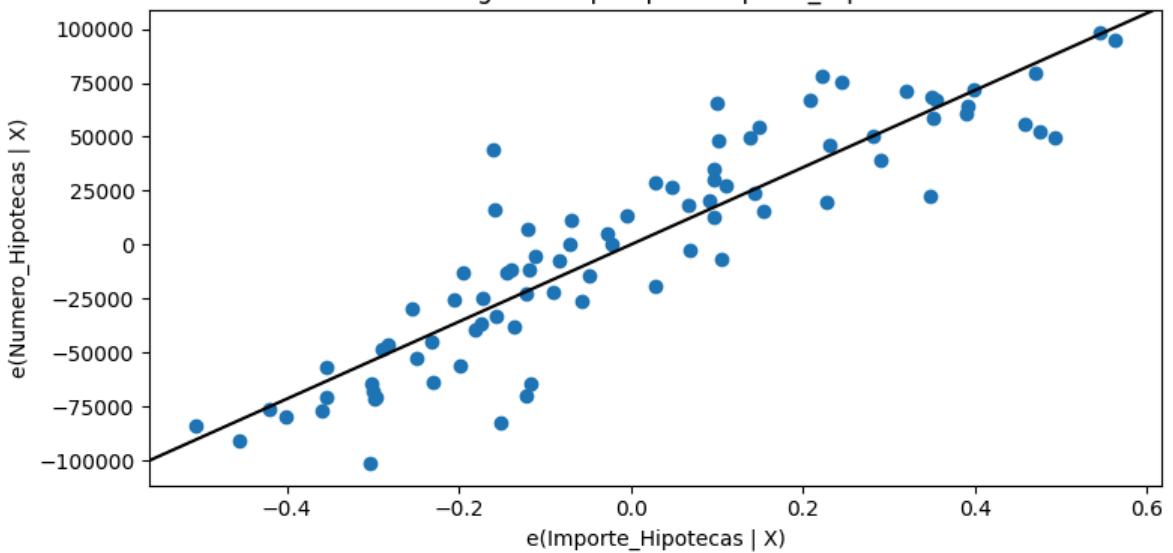
if len(independent_vars_temp3) == 1:
    axes = [axes]

for ax, var in zip(axes, independent_vars_temp3):
    # Generamos partial regression plot
    plot_partregress(endog="Numero_Hipotecas",
                      exog_i=var,
                      exog_others=[v for v in independent_vars_temp3 if v != var],
                      data=df_temp3, ax=ax)
    ax.set_title(f'Partial regression plot para {var}')

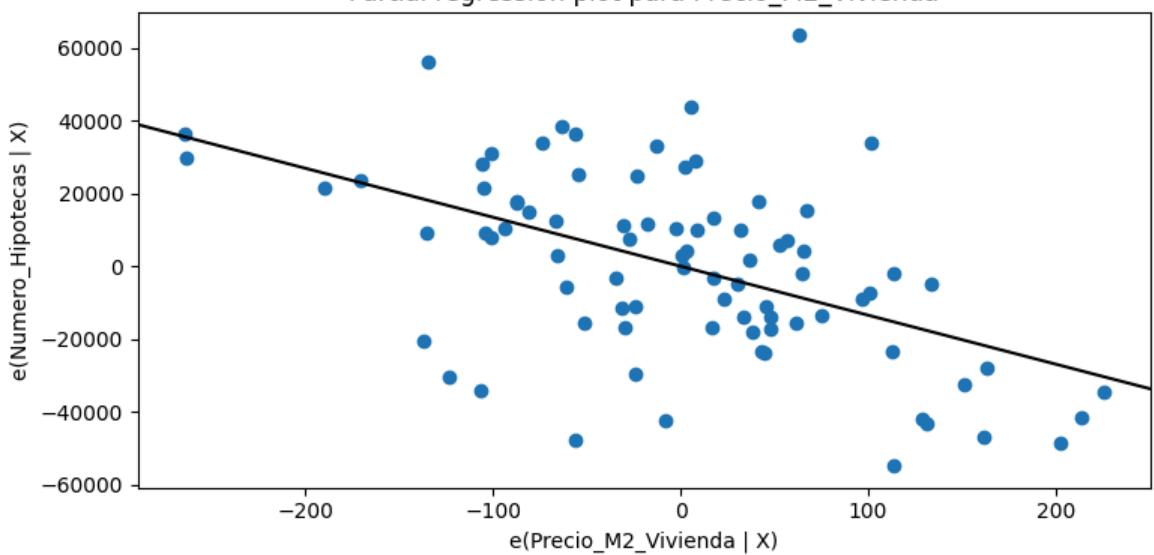
    # Quitar las etiquetas de puntos (textos)
    for txt in ax.texts:
        txt.set_visible(False)
```

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

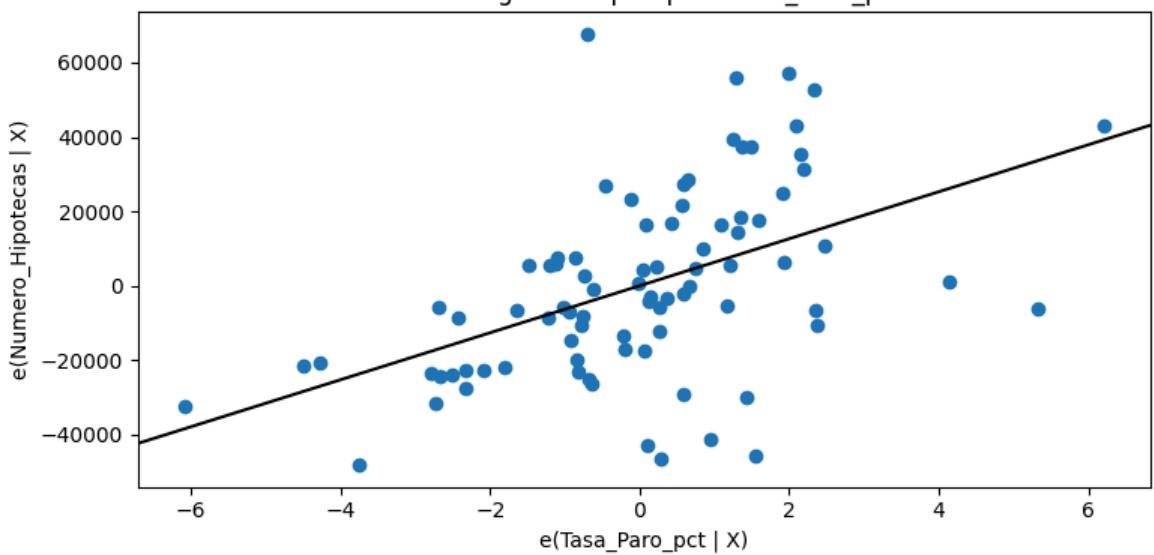
Partial regression plot para Importe_Hipotecas



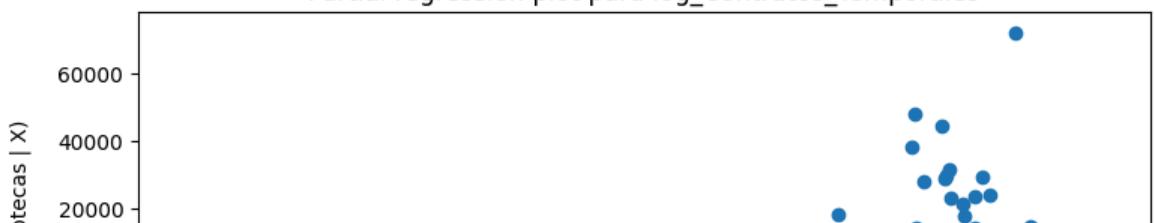
Partial regression plot para Precio_M2_Vivienda

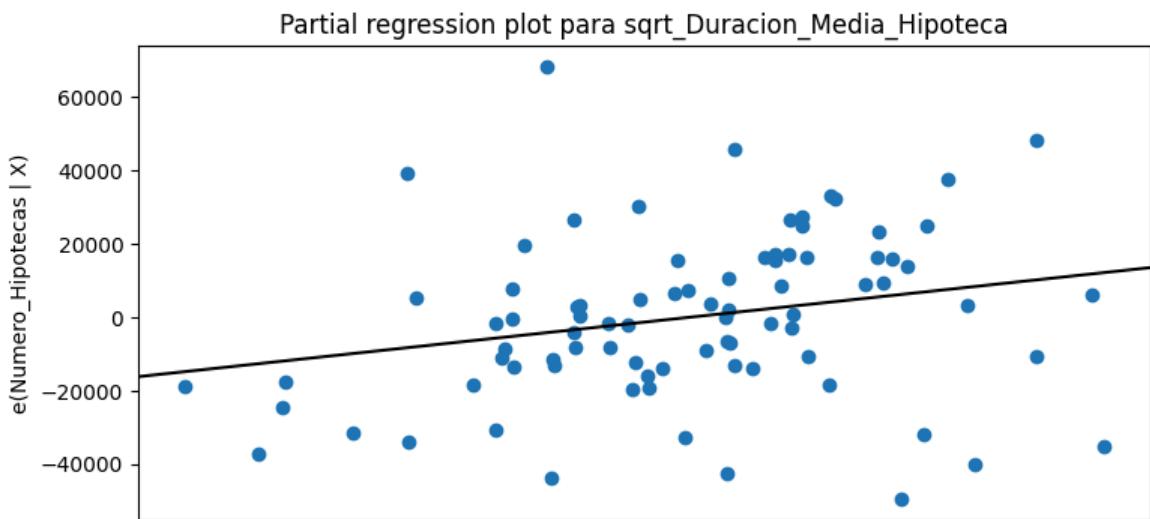
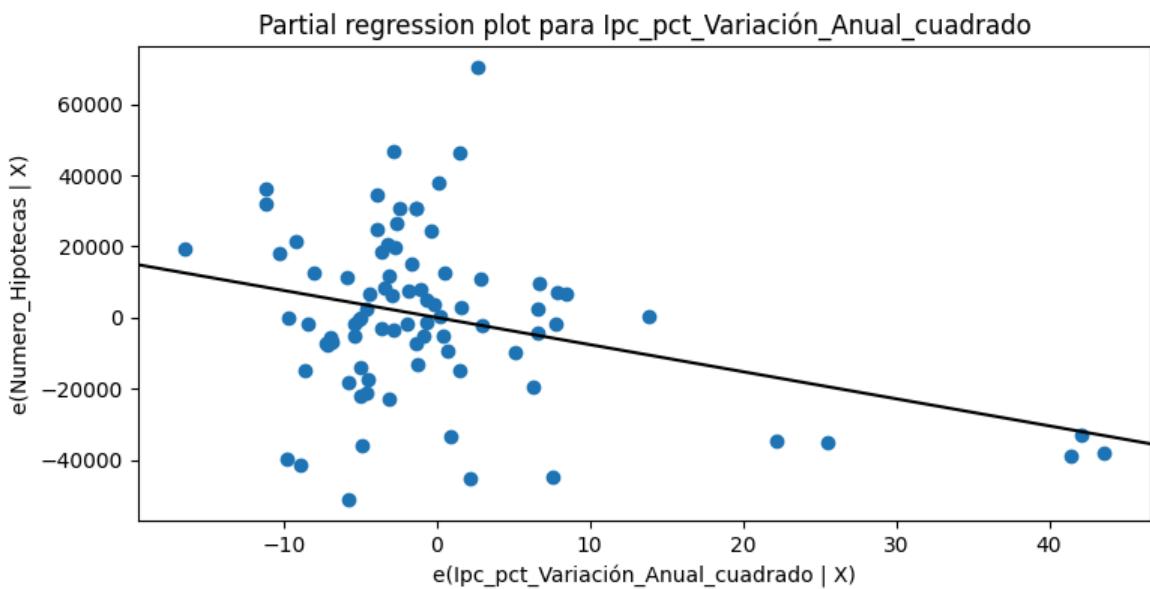
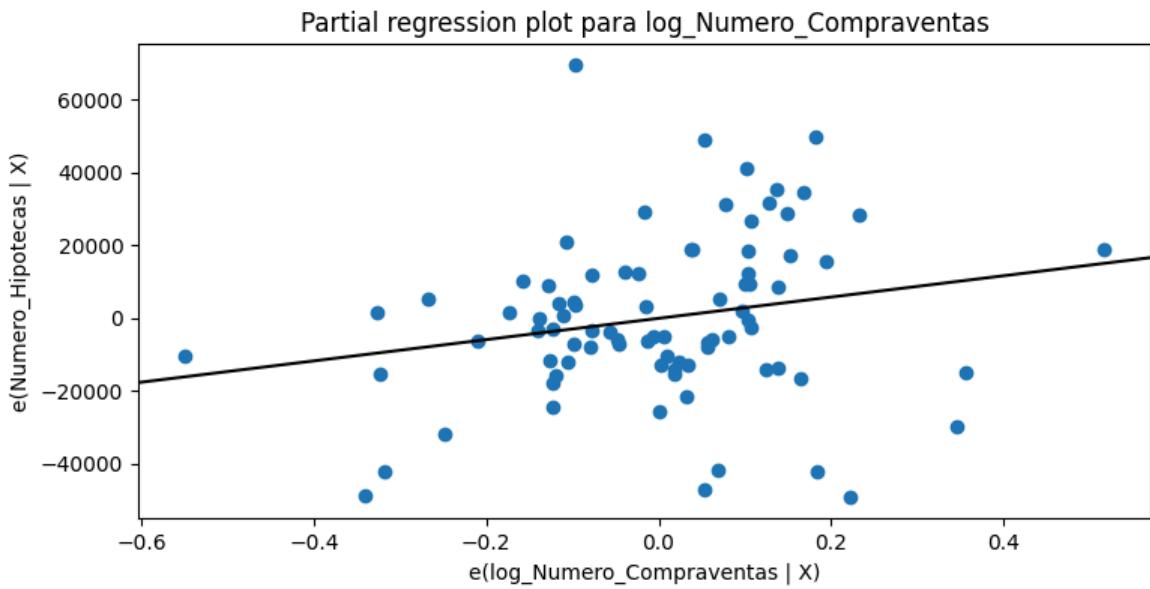
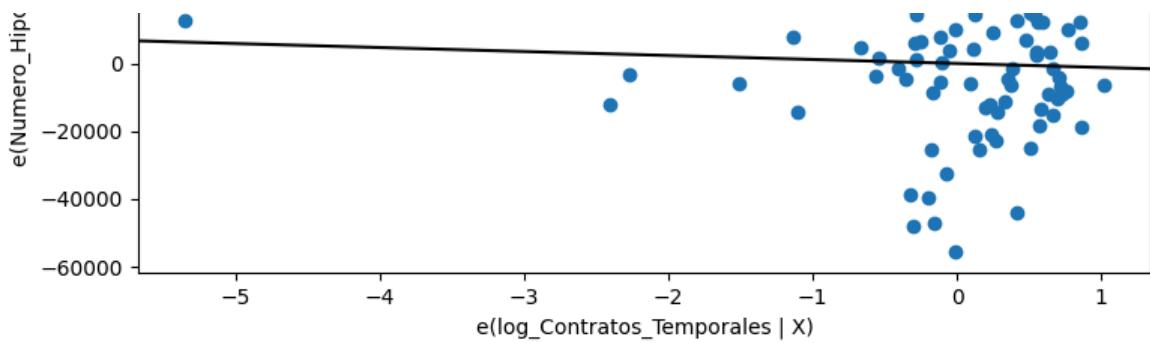


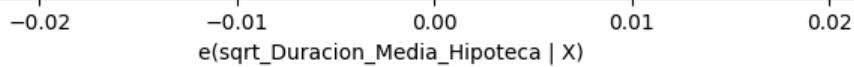
Partial regression plot para Tasa_Paro_pct



Partial regression plot para log_Contratos_Temporales







Comprobamos que, aplicando las transformaciones, **sigue sin cumplirse el supuesto de A) Linealidad de los parámetros**, por lo que lo más probable es que lo óptimo sea utilizar otro tipo de modelo que no sea lineal (limitación del modelo de regresión lineal múltiple).

Aún así, procedemos a realizar una predicción con dicho modelo para evaluar en qué medida predice correctamente los datos de hipotecas concedidas de 2023. Sin embargo, por las razones que hemos expuesto, es muy probable que el modelo lineal ajuste peor, que los coeficientes sean sesgados o que pierdan potencia estadística.

En definitiva, **los coeficientes obtenidos se deben interpretar con cautela**. No obstante, a veces el modelo global puede predecir bien (aunque se reduzca la validez a la hora de inferir el efecto de cada variable en la variable objetivo).

Paso 3: Predicciones

A) Predicción sobre df_no_corr sin transformación de variables no lineales

```
In [27]: # Aseguramos que el índice es datetime
df.index = pd.to_datetime(df.index)

# --- Datos entrenamiento (antes de 2023) ---
serie_train = df[df.index.year < 2023]['Número_Hipotecas']

# --- Preparar datos de 2023 ---
X_2023 = df[df.index.year == 2023].drop(columns=['Número_Hipotecas'])
y_2023 = df[df.index.year == 2023]['Número_Hipotecas']
X_2023 = sm.add_constant(X_2023)

# --- Predicciones ---
y_pred_2023 = model.predict(X_2023)

# --- Evaluación ---
mae = mean_absolute_error(y_2023, y_pred_2023)
rmse = np.sqrt(mean_squared_error(y_2023, y_pred_2023))
r2 = r2_score(y_2023, y_pred_2023)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.2f}")

# --- Comparativa gráfica estilo SARIMA ---

plt.figure(figsize=(12,5))

# Serie entrenamiento
plt.plot(serie_train, label="Entrenamiento (antes 2023)")

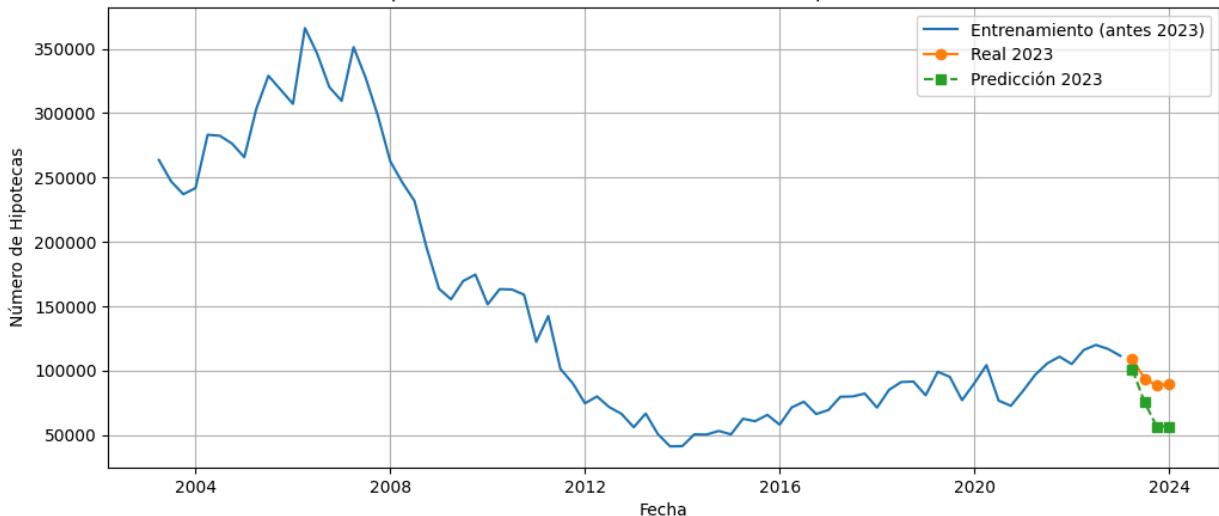
# Serie real 2023
plt.plot(y_2023.index, y_2023.values, label="Real 2023", marker='o')

# Predicción 2023 (línea discontinua y marcador distinto)
plt.plot(y_2023.index, y_pred_2023.values, label="Predicción 2023", linestyle='--', marker='s')

plt.title("Comparativa Predicción vs Real - Número de Hipotecas en 2023")
plt.xlabel("Fecha")
plt.ylabel("Número de Hipotecas")
plt.legend()
plt.grid(True)
plt.show()

MAE: 22884.61
RMSE: 25110.96
R²: -8.41
```

Comparativa Predicción vs Real - Número de Hipotecas en 2023



```
In [28]: # Aseguramos que el índice es datetime
df.index = pd.to_datetime(df.index)

# --- Preparar datos de 2023 ---
X_2023 = df[df.index.year == 2023].drop(columns=['Número_Hipotecas'])
y_2023 = df[df.index.year == 2023]['Número_Hipotecas']
X_2023 = sm.add_constant(X_2023)

# --- Predicciones ---
y_pred_2023 = model.predict(X_2023)

# --- Evaluación ---
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_2023, y_pred_2023)
rmse = np.sqrt(mean_squared_error(y_2023, y_pred_2023))
r2 = r2_score(y_2023, y_pred_2023)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.2f}")

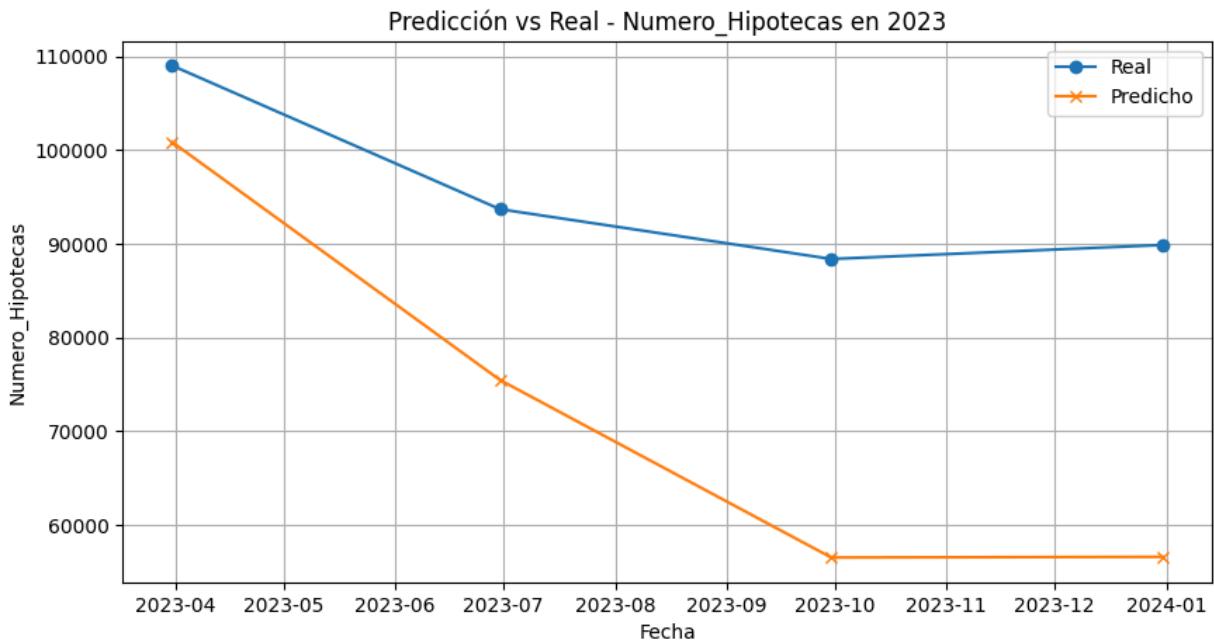
# --- Comparativa gráfica ---

plt.figure(figsize=(10,5))
plt.plot(y_2023.index, y_2023.values, label='Real', marker='o')
plt.plot(y_2023.index, y_pred_2023.values, label='Predicho', marker='x')
plt.title('Predicción vs Real - Número_Hipotecas en 2023')
plt.xlabel('Fecha')
plt.ylabel('Número_Hipotecas')
plt.legend()
plt.grid()
plt.show()
```

MAE: 22884.61

RMSE: 25110.96

R²: -8.41



```
In [29]: df_diferencias = pd.DataFrame({
    'Real': y_2023,
    'Predicho': y_pred_2023,
    'Diferencia': y_2023 - y_pred_2023
})

print(df_diferencias)
```

Fecha	Real	Predicho	Diferencia
2023-03-31	109020	100860.156210	8159.843790
2023-06-30	93695	75469.953935	18225.046065
2023-09-30	88378	56513.816158	31864.183842
2023-12-31	89873	56583.652900	33289.347100

```
In [30]: # Diferencia total absoluta (suma de |real - predicho|)
diferencia_absoluta = (y_2023 - y_pred_2023).abs().sum()
print(f"Diferencia absoluta total: {diferencia_absoluta:.2f}")

Diferencia absoluta total: 91538.42
```

B) Predicción sobre df_no_corr con transformación de variables no lineales

```
In [31]: # Aseguramos que el índice es datetime
df3.index = pd.to_datetime(df3.index)

# --- Datos entrenamiento (antes de 2023) ---
serie_train = df3[df3.index.year < 2023]['Número_Hipotecas']

# --- Preparar datos de 2023 ---
X_2023 = df3[df3.index.year == 2023].drop(columns=['Número_Hipotecas'])
y_2023 = df3[df3.index.year == 2023]['Número_Hipotecas']
X_2023 = sm.add_constant(X_2023)

# --- Predicciones ---
y_pred_2023 = model3.predict(X_2023)

# --- Evaluación ---
mae = mean_absolute_error(y_2023, y_pred_2023)
rmse = np.sqrt(mean_squared_error(y_2023, y_pred_2023))
r2 = r2_score(y_2023, y_pred_2023)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R2: {r2:.2f}")

# --- Comparativa gráfica estilo SARIMA ---

plt.figure(figsize=(12,5))

# Serie entrenamiento
plt.plot(serie_train, label="Entrenamiento (antes 2023)")
```

```

# Serie real 2023
plt.plot(y_2023.index, y_2023.values, label="Real 2023", marker='o')

# Predicción 2023 (Línea discontinua y marcador distinto)
plt.plot(y_2023.index, y_pred_2023.values, label="Predicción 2023", linestyle='--', marker='s')

plt.title("Comparativa Predicción vs Real - Número de Hipotecas en 2023")
plt.xlabel("Fecha")
plt.ylabel("Número de Hipotecas")
plt.legend()
plt.grid(True)
plt.show()

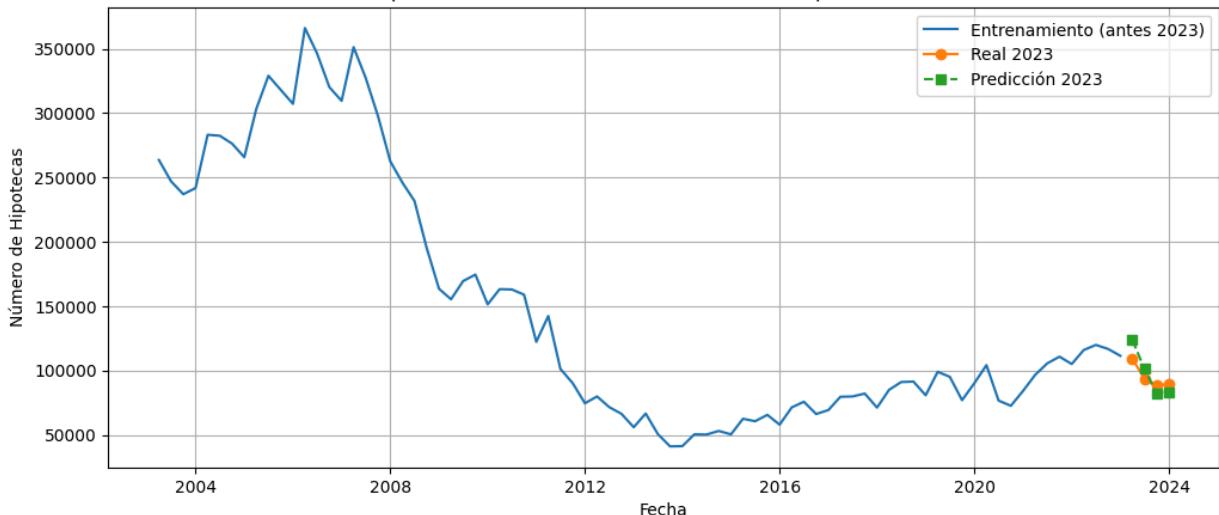
```

MAE: 8795.90

RMSE: 9454.46

R²: -0.33

Comparativa Predicción vs Real - Número de Hipotecas en 2023



```

In [32]: # Aseguramos que el índice es datetime
df3.index = pd.to_datetime(df3.index)

# --- Preparar datos de 2023 ---
X_2023 = df3[df3.index.year == 2023].drop(columns=['Número_Hipotecas'])
y_2023 = df3[df3.index.year == 2023]['Número_Hipotecas']
X_2023 = sm.add_constant(X_2023)

# --- Predicciones ---
y_pred_2023 = model3.predict(X_2023)

# --- Evaluación ---
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_2023, y_pred_2023)
rmse = np.sqrt(mean_squared_error(y_2023, y_pred_2023))
r2 = r2_score(y_2023, y_pred_2023)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R2: {r2:.2f}")

# --- Comparativa gráfica ---

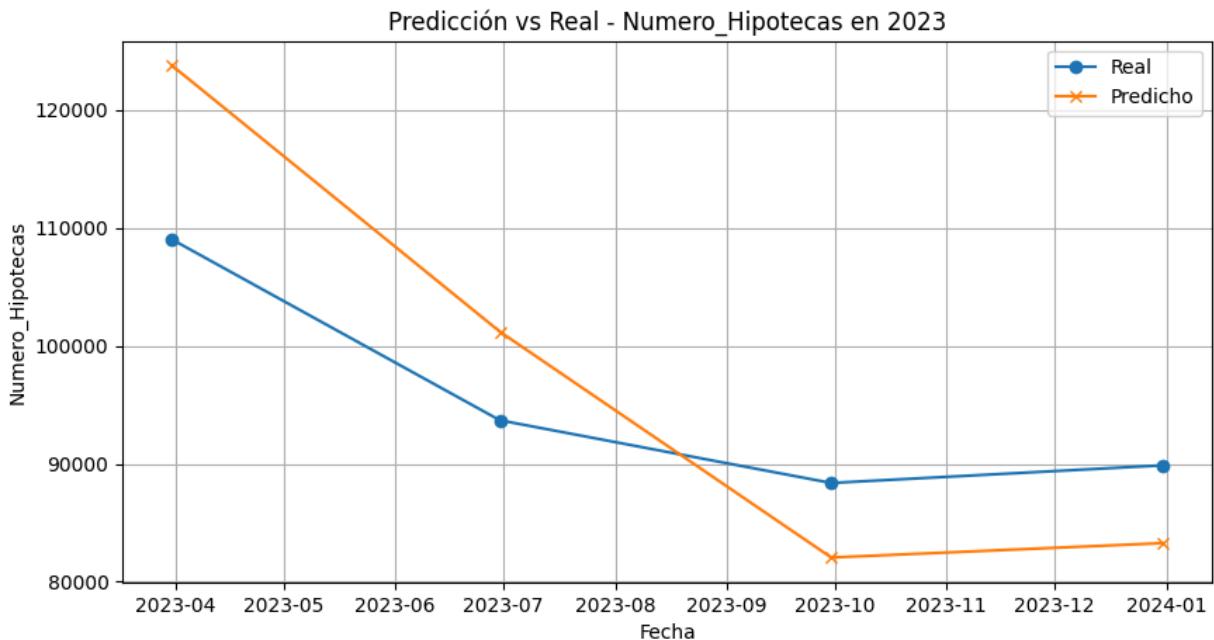
plt.figure(figsize=(10,5))
plt.plot(y_2023.index, y_2023.values, label='Real', marker='o')
plt.plot(y_2023.index, y_pred_2023.values, label='Predicho', marker='x')
plt.title('Predicción vs Real - Número_Hipotecas en 2023')
plt.xlabel('Fecha')
plt.ylabel('Número_Hipotecas')
plt.legend()
plt.grid()
plt.show()

```

MAE: 8795.90

RMSE: 9454.46

R²: -0.33



```
In [33]: df_diferencias2 = pd.DataFrame({
    'Real': y_2023,
    'Predicho': y_pred_2023,
    'Diferencia': y_2023 - y_pred_2023
})
```

```
Real      Predicho      Diferencia
Fecha
2023-03-31  109020  123771.706423 -14751.706423
2023-06-30   93695  101208.999613 -7513.999613
2023-09-30   88378  82056.847420  6321.152580
2023-12-31   89873  83276.254000  6596.746000
```

```
In [34]: # Diferencia total absoluta (suma de |real - predicho|)
diferencia_absoluta = (y_2023 - y_pred_2023).abs().sum()
print(f"Diferencia absoluta total: {diferencia_absoluta:.2f}")
```

```
Diferencia absoluta total: 35183.60
```

C) Predicción sobre iqr_winsorized_df con transformación de variables no lineales

```
In [35]: # Aseguramos que el índice es datetime
df2.index = pd.to_datetime(df2.index)

# Columnas que usó el modelo para entrenar (incluida la constante)
cols_modelo = model2.model.exog_names

# --- Datos entrenamiento (antes de 2023) ---
serie_train = df2[df2.index.year < 2023]['Número_Hipotecas']

# --- Preparar datos de 2023 ---
X_2023 = df2[df2.index.year == 2023].drop(columns=['Número_Hipotecas'])

# Añadimos constante (asegurando que solo haya una)
X_2023 = sm.add_constant(X_2023, has_constant='add')

# Añadir columnas faltantes con ceros
for col in cols_modelo:
    if col not in X_2023.columns:
        X_2023[col] = 0

# Reordenar columnas para que coincidan con el modelo
X_2023 = X_2023[cols_modelo]

# Variable objetivo 2023
y_2023 = df2[df2.index.year == 2023]['Número_Hipotecas']

# --- Predicciones ---
y_pred_2023 = model2.predict(X_2023)

# --- Evaluación ---
```

```

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_2023, y_pred_2023)
rmse = np.sqrt(mean_squared_error(y_2023, y_pred_2023))
r2 = r2_score(y_2023, y_pred_2023)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.2f}")

# --- Comparativa gráfica estilo SARIMA ---

plt.figure(figsize=(12,5))

# Serie entrenamiento
plt.plot(serie_train, label="Entrenamiento (antes 2023)")

# Serie real 2023
plt.plot(y_2023.index, y_2023.values, label="Real 2023", marker='o')

# Predicción 2023 (Línea discontinua y marcador distinto)
plt.plot(y_2023.index, y_pred_2023.values, label="Predicción 2023", linestyle='--', marker='s')

plt.title("Comparativa Predicción vs Real - Número de Hipotecas en 2023")
plt.xlabel("Fecha")
plt.ylabel("Número de Hipotecas")
plt.legend()
plt.grid(True)
plt.show()

```

MAE: 4844.31

RMSE: 5379.12

R²: 0.57

Comparativa Predicción vs Real - Número de Hipotecas en 2023



```

In [36]: # Aseguramos que el índice es datetime
df2.index = pd.to_datetime(df2.index)

# Columnas que usó el modelo para entrenar (incluida la constante)
cols_modelo = modelo2.model.exog_names # Lista de columnas que el modelo espera

# --- Preparar datos de 2023 ---
X_2023 = df2[df2.index.year == 2023].drop(columns=['Número_Hipotecas'])

# Añadimos constante (asegurándonos que solo haya una)
X_2023 = sm.add_constant(X_2023, has_constant='add')

# Añadir columnas faltantes con ceros
for col in cols_modelo:
    if col not in X_2023.columns:
        X_2023[col] = 0

# Reordenar columnas para que coincidan exactamente con las del modelo
X_2023 = X_2023[cols_modelo]

# Variable objetivo para 2023
y_2023 = df2[df2.index.year == 2023]['Número_Hipotecas']

# --- Predicciones ---

```

```

y_pred_2023 = model2.predict(X_2023)

# --- Evaluación ---
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_2023, y_pred_2023)
rmse = np.sqrt(mean_squared_error(y_2023, y_pred_2023))
r2 = r2_score(y_2023, y_pred_2023)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.2f}")

# --- Comparativa gráfica ---

plt.figure(figsize=(10,5))
plt.plot(y_2023.index, y_2023.values, label='Real', marker='o')
plt.plot(y_2023.index, y_pred_2023.values, label='Predicho', marker='x')
plt.title('Predicción vs Real - Número_Hipotecas en 2023')
plt.xlabel('Fecha')
plt.ylabel('Número_Hipotecas')
plt.legend()
plt.grid()
plt.show()
y_pred_2023 = model2.predict(X_2023)

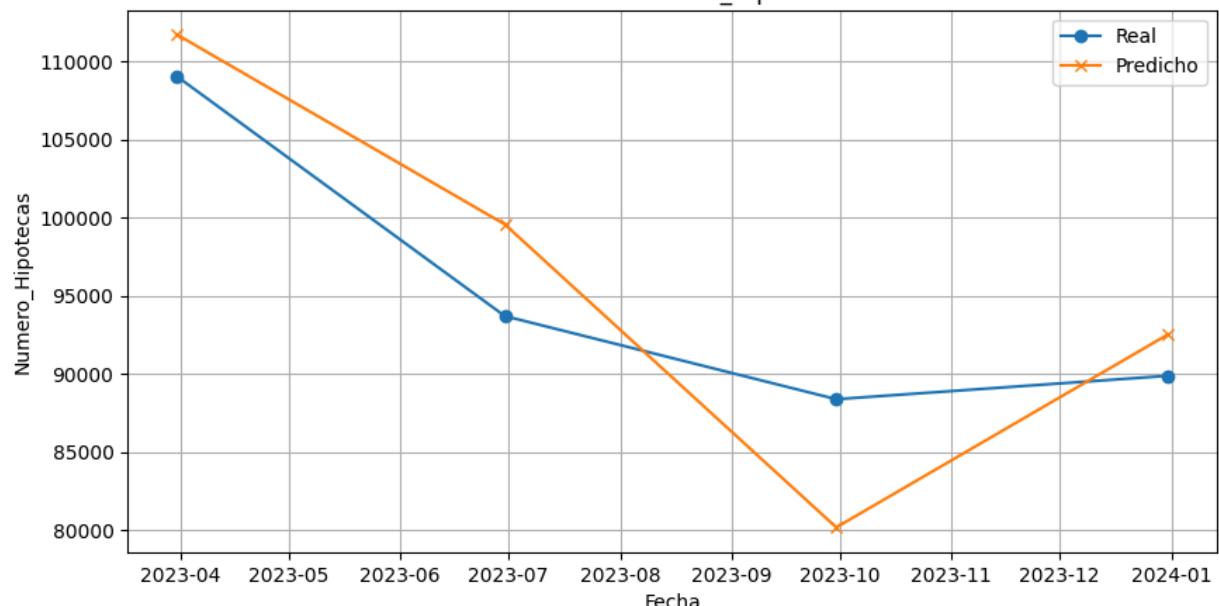
```

MAE: 4844.31

RMSE: 5379.12

R²: 0.57

Predicción vs Real - Número_Hipotecas en 2023



```

In [37]: df_diferencias3 = pd.DataFrame({
    'Real': y_2023,
    'Predicho': y_pred_2023,
    'Diferencia': y_2023 - y_pred_2023
})

print(df_diferencias3)

```

Fecha	Real	Predicho	Diferencia
2023-03-31	109020	111696.685140	-2676.685140
2023-06-30	93695	99559.636311	-5864.636311
2023-09-30	88378	80178.603024	8199.396976
2023-12-31	89873	92509.511736	-2636.511736

```

In [38]: # Diferencia total absoluta (suma de |real - predicho|)
diferencia_absoluta = (y_2023 - y_pred_2023).abs().sum()
print(f"Diferencia absoluta total: {diferencia_absoluta:.2f}")

```

Diferencia absoluta total: 19377.23

Paso 4: Resultados y conclusiones

Análisis por modelo

Modelo 1 (df_no_corr sin transformaciones)

- Métricas malas (R^2 negativo muy alto), error absoluto muy alto.
- No se cumplen dos supuestos clave: linealidad y no autocorrelación.
- El modelo no funciona bien para predecir la variable objetivo, probablemente subajustado o mal especificado.
- El error total de predicción es enorme (más de 100k hipotecas de diferencia total), por lo que no es útil.

Modelo 2 (df_no_corr con transformaciones)

- Métricas mejoran mucho respecto al Modelo 1, pero siguen siendo malas: R^2 negativo (-1.28) indica que aún predice peor que la media.
- Los supuestos problemáticos son los mismos que en el Modelo 1, indicando que la estructura del modelo sigue sin ajustarse bien a la realidad.
- El error total absoluto baja bastante (casi 42k hipotecas), pero sigue siendo elevado.
- Mejor que Modelo 1, pero aún insuficiente.

Modelo 3 (iqr_winsorized_df con todas variables)

- Métricas muy buenas: $R^2 = 0.89$ indica que el modelo explica el 89% de la variabilidad.
- MAE y RMSE mucho más bajos que en los otros modelos.
- El error total absoluto es también mucho menor (9,850 hipotecas).
- No se cumplen algunos supuestos (linealidad, multicolinealidad perfecta y esperanza del error), lo que puede afectar la interpretabilidad y estabilidad del modelo, pero estadísticamente es mucho más sólido para predicción.
- Posible problema con multicolinealidad y residuos, pero el poder predictivo es claramente superior.

¿Cuál es el mejor modelo? Claramente el Modelo 3, a pesar de que no cumple todos los supuestos clásicos, ofrece un ajuste mucho mejor a los datos y predice la variable objetivo con mucha mayor precisión:

- Tiene un R^2 positivo y alto (0.89), mientras los otros dos son negativos.
- El MAE y RMSE son mucho más bajos, mostrando predicciones más precisas.
- El error total absoluto es muy inferior, por lo que la suma de las diferencias absolutas entre real y predicho es mucho menor.
- El problema de los supuestos no cumplidos (como la multicolinealidad) puede manejarse con técnicas adicionales si fuera necesario (p.ej., regularización, selección de variables, técnicas robustas). Pero en cuanto a capacidad predictiva pura, es claramente el mejor.

La baja calidad de los modelos 1 y 2 y el R^2 negativo indican que el modelo está mal especificado o que el dataset sin variables correlacionadas no es suficiente para capturar la dinámica de la variable objetivo.

2. ARIMA/SARIMA

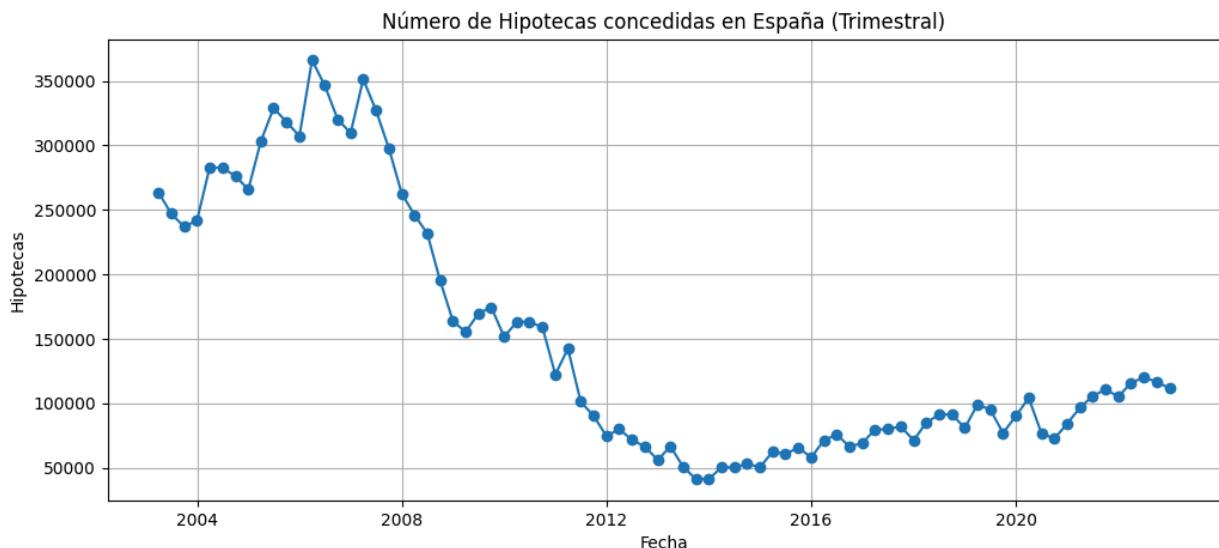
```
In [39]: # Cargamos el dataset
df = pd.read_excel("df_no_corr.xlsx", sheet_name="variables")

# Nos aseguramos de que la columna 'Fecha' esté en formato datetime y como índice
df["Fecha"] = pd.to_datetime(df["Fecha"])
df.set_index("Fecha", inplace=True)

# Nos quedamos solo con la columna objetivo para el modelo ARIMA/SARIMA
serie = df["Número_Hipotecas"]
# Entrenamos de 2003 a 2022, test 2023
serie_train = serie.loc[:'2022-12-31']
serie_test = serie.loc['2023-03-31':'2023-12-31']
```

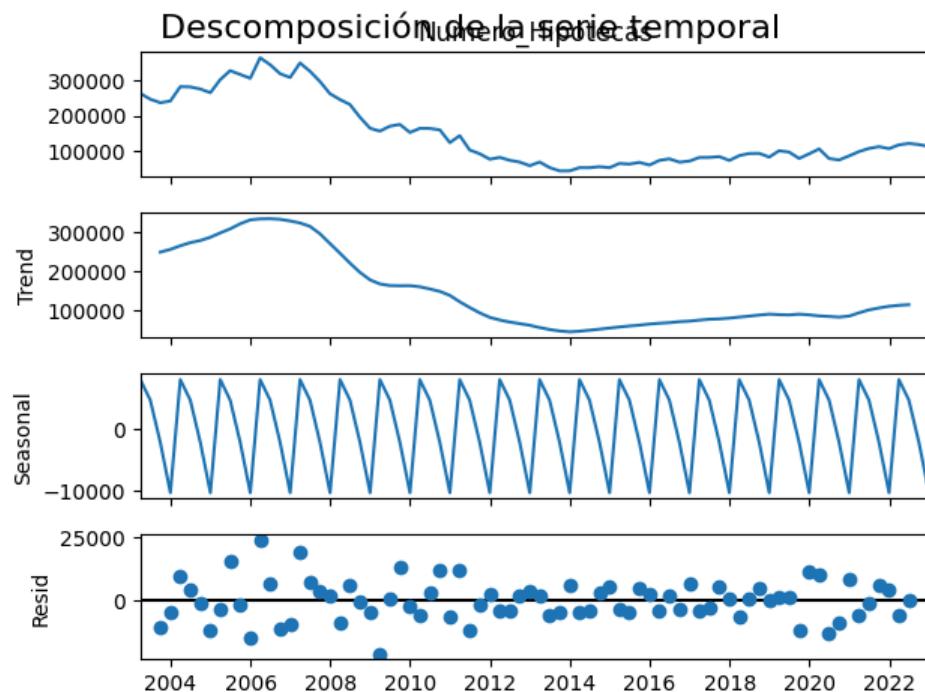
Paso 2: Visualización y análisis exploratorio

```
In [40]: plt.figure(figsize=(12, 5))
plt.plot(serie_train, marker='o')
plt.title("Número de Hipotecas concedidas en España (Trimestral)")
plt.xlabel("Fecha")
plt.ylabel("Hipotecas")
plt.grid(True)
plt.show()
```



Paso 3: Comprobación de estacionalidad y tendencia

```
In [41]: result = seasonal_decompose(serie_train, model='additive', period=4) # 4 trimestres por año
result.plot()
plt.suptitle("Descomposición de la serie temporal", fontsize=16)
plt.show()
```



Paso 4: Test de estacionariedad (Dickey-Fuller)

```
In [42]: adf_result = adfuller(serie_train)
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])
```

```
ADF Statistic: -1.9751590897572295
p-value: 0.29757004427355616
```

El p-value es mayor a 0.05, lo que indica que la serie no es estacionaria. Por tanto, es necesario aplicar una diferenciación para estabilizar la media y permitir el uso de modelos ARIMA/SARIMA.

Paso 5: Diferenciación de la serie y nuevo test ADF

```
In [43]: # Diferenciamos la serie
serie_diff = serie_train.diff().dropna()

# Repetimos el test ADF
adf_diff = adfuller(serie_diff)

print("ADF Statistic (d=1):", adf_diff[0])
print("p-value:", adf_diff[1])
```

ADF Statistic (d=1): -2.316503635126042
p-value: 0.16670045309658327

El p-value sigue siendo mayor a 0.05, por lo que no se puede rechazar la hipótesis nula de que la serie aún no es estacionaria. Esto significa que una sola diferenciación no es suficiente, y por tanto vamos a aplicar una segunda diferenciación (d=2), algo que en datos macroeconómicos de largo plazo no es inusual, especialmente cuando hay una combinación de tendencia y estacionalidad fuerte

Segunda Diferenciación

```
In [44]: # Segunda diferenciación
serie_diff2 = serie_train.diff().diff().dropna()

# Test ADF sobre la segunda diferenciación
adf_diff2 = adfuller(serie_diff2)

print("ADF Statistic (d=2):", adf_diff2[0])
print("p-value:", adf_diff2[1])
```

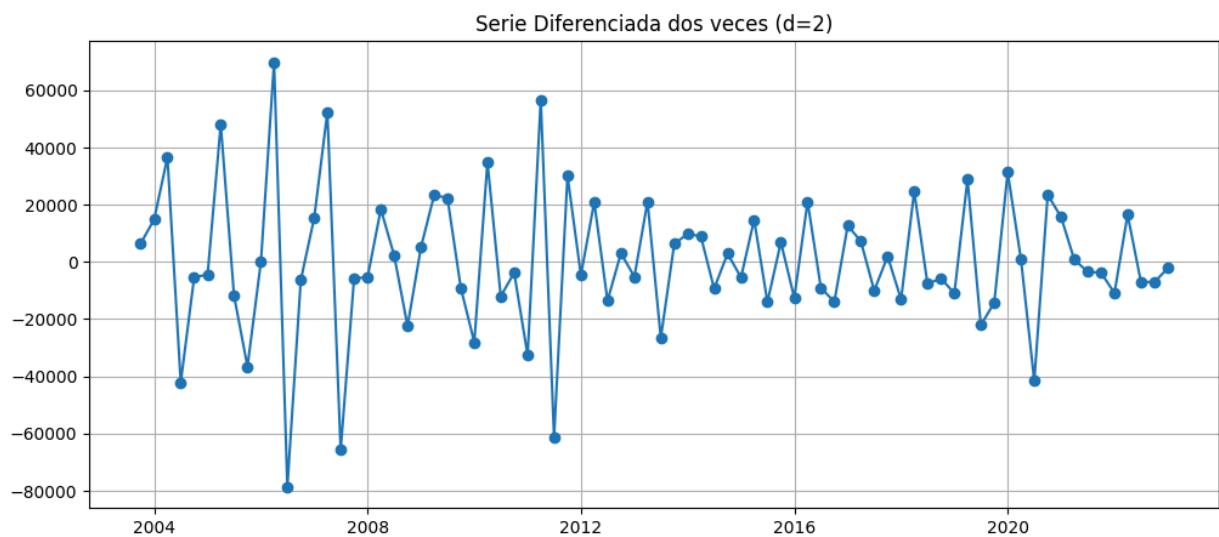
ADF Statistic (d=2): -15.784933470597675
p-value: 1.1197138394909665e-28

El p-value es prácticamente cero, muy por debajo del umbral del 0.05.

Por tanto, la serie diferenciada dos veces es estacionaria.

Podemos proceder a ajustar un modelo SARIMA con d=2 para capturar tanto las dinámicas temporales como la estacionalidad trimestral.

```
In [45]: plt.figure(figsize=(12, 5))
plt.plot(serie_diff2, marker='o')
plt.title("Serie Diferenciada dos veces (d=2)")
plt.grid(True)
plt.show()
```



Paso 7: Ajuste del modelo SARIMA

```
In [46]: # Entrenamiento del modelo SARIMA (d=2, estacionalidad trimestral s=4)
modelo_sarima = SARIMAX(serie_train,
                        order=(1,2,1),
                        seasonal_order=(1,1,1,4),
                        enforce_stationarity=False,
                        enforce_invertibility=False)

resultado = modelo_sarima.fit()
print(resultado.summary())
```

```
SARIMAX Results
=====
Dep. Variable: Numero_Hipotecas No. Observations: 80
Model: SARIMAX(1, 2, 1)x(1, 1, 4) Log Likelihood: -757.064
Date: Wed, 09 Jul 2025 AIC: 1524.127
Time: 09:27:12 BIC: 1535.225
Sample: 03-31-2003 HQIC: 1528.524
- 12-31-2022
Covariance Type: opg
=====
            coef    std err        z    P>|z|    [0.025    0.975]
-----
ar.L1     -0.1287    0.229   -0.561    0.574    -0.578    0.321
ma.L1     -0.6830    0.228   -2.995    0.003    -1.130    -0.236
ar.S.L4    -0.0931    0.172   -0.543    0.587    -0.429    0.243
ma.S.L4    -0.4152    0.229   -1.813    0.070    -0.864    0.034
sigma2    3.738e+08  5.58e-10  6.7e+17    0.000    3.74e+08  3.74e+08
=====
Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 15.47
Prob(Q): 0.98 Prob(JB): 0.00
Heteroskedasticity (H): 0.31 Skew: -0.11
Prob(H) (two-sided): 0.01 Kurtosis: 5.33
=====
```

Warnings:

```
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 6.88e+33. Standard errors may be unstable.
```

```
C:\Users\luife\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueError
Warning: No frequency information was provided, so inferred frequency QE-DEC will be used.
    self._init_dates(dates, freq)
C:\Users\luife\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueError
Warning: No frequency information was provided, so inferred frequency QE-DEC will be used.
    self._init_dates(dates, freq)
```

Paso 8: Predicción y evaluación del modelo SARIMA

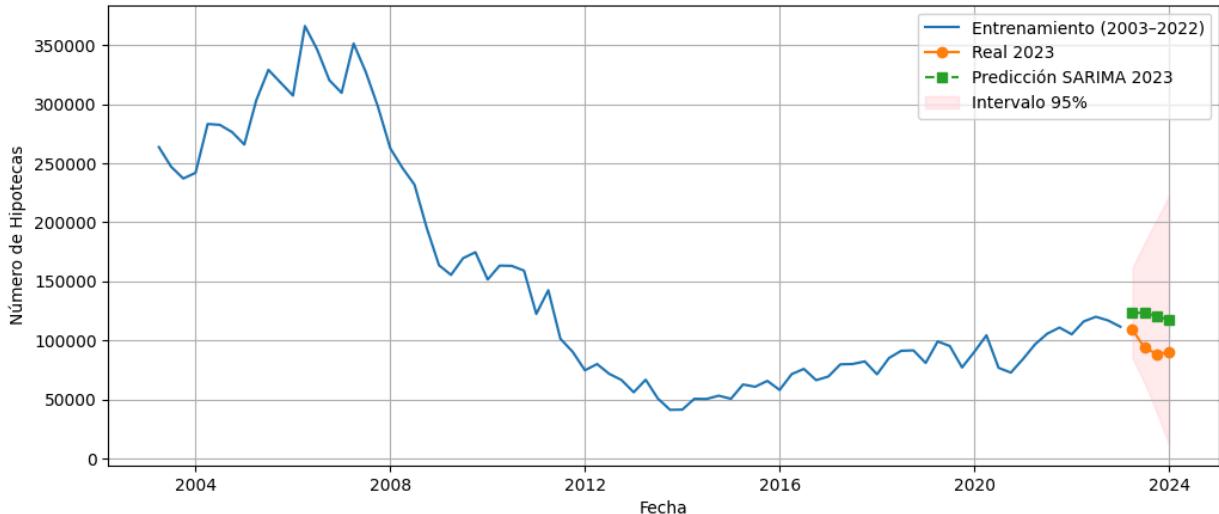
```
In [47]: forecast_sarima = resultado.get_forecast(steps=4)
pred_2023_sarima = forecast_sarima.predicted_mean

# Asignamos las fechas correspondientes a los trimestres de 2023
pred_2023_sarima.index = pd.to_datetime(['2023-03-31', '2023-06-30', '2023-09-30', '2023-12-31'])

# Intervalo de confianza del 95%
conf_int = forecast_sarima.conf_int()
conf_int.index = pred_2023_sarima.index

plt.figure(figsize=(12,5))
plt.plot(serie_train, label="Entrenamiento (2003-2022)")
plt.plot(serie_test, label="Real 2023", marker='o')
plt.plot(pred_2023_sarima, label="Predicción SARIMA 2023", linestyle='--', marker='s')
plt.fill_between(conf_int.index, conf_int.iloc[:, 0], conf_int.iloc[:, 1],
                 color='pink', alpha=0.3, label='Intervalo 95%')
plt.title("Predicción del número de hipotecas en 2023 con modelo SARIMA")
plt.xlabel("Fecha")
plt.ylabel("Número de Hipotecas")
plt.legend()
plt.grid(True)
plt.show()
```

Predicción del número de hipotecas en 2023 con modelo SARIMA



```
In [48]: # Cálculo de métricas de evaluación

# Valores reales del test
real_2023 = serie_test

# Cálculo de errores
rmse = np.sqrt(mean_squared_error(real_2023, pred_2023_sarima))
mae = mean_absolute_error(real_2023, pred_2023_sarima)

print(f"SARIMA (predicción 2023) -> RMSE: {rmse:.2f} | MAE: {mae:.2f}")

SARIMA (predicción 2023) -> RMSE: 26700.33 | MAE: 25787.33
```

```
In [49]: comparacion_sarima = pd.DataFrame({
    "Hipotecas Reales": real_2023.values,
    "Predicción SARIMA": pred_2023_sarima,
    "Error Absoluto": np.abs(real_2023.values - pred_2023_sarima)
}, index=real_2023.index)

print(comparacion_sarima.round(0))
```

Fecha	Hipotecas Reales	Predicción SARIMA	Error Absoluto
2023-03-31	109020	123237.0	14217.0
2023-06-30	93695	123183.0	29488.0
2023-09-30	88378	120665.0	32287.0
2023-12-31	89873	117030.0	27157.0

3. SARIMAX

```
In [50]: # Creamos copia del DataFrame original
df_model = df.copy()

# Variable dependiente (target)
y = df_model["Número_Hipotecas"]

# Variables exógenas: eliminamos columnas no útiles y la variable objetivo
exog = df_model.drop(columns=["Número_Hipotecas"])
```

Paso 2: División temporal en train y test

```
In [51]: # Train: 2003 - 2022 / Test: 2023
y_train = y.loc[:'2022-12-31']
y_test = y.loc['2023-03-31':'2023-12-31']

exog_train = exog.loc[:'2022-12-31']
exog_test = exog.loc['2023-03-31':'2023-12-31']
```

Paso 3: Entrenamiento de SARIMAX

```
In [52]: # Definimos el modelo SARIMAX
modelo_sarimax = SARIMAX(endog=y_train,
```

```

        exog=exog_train,
        order=(1,1,1), # configuración inicial
        seasonal_order=(1,1,1,4),
        enforce_stationarity=False,
        enforce_invertibility=False)

resultado_sarimax = modelo_sarimax.fit()
print(resultado_sarimax.summary())

```

```

C:\Users\luife\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency QE-DEC will be used.
  self._init_dates(dates, freq)
C:\Users\luife\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency QE-DEC will be used.
  self._init_dates(dates, freq)

SARIMAX Results
=====
Dep. Variable:           Numero_Hipotecas    No. Observations:                  80
Model:                 SARIMAX(1, 1, 1)x(1, 1, 4)    Log Likelihood:                -717.730
Date:                  Wed, 09 Jul 2025   AIC:                            1459.461
Time:                  09:27:13           BIC:                            1486.270
Sample:                03-31-2003   HQIC:                           1470.097
                           - 12-31-2022
Covariance Type:            opg
=====
              coef    std err        z      P>|z|      [0.025      0.975]
-----
Importe_Hipotecas    9.575e+04    9172.127    10.439      0.000    7.78e+04    1.14e+05
Numero_Compraventas    0.1117      0.052      2.153      0.031      0.010      0.213
Precio_M2_Vivienda    -47.9641    105.494     -0.455      0.649    -254.728    158.800
Contratos_Temporales    -11.3168      9.094     -1.244      0.213     -29.140      6.506
Ipc_%_Variación_Annual    -1669.5684   1734.163     -0.963      0.336    -5068.465    1729.328
Duracion_Media_Hipoteca    8736.8394   4.98e+04     0.175      0.861    -8.89e+04    1.06e+05
Tasa_Paro_(%)    -2385.2290    2885.046     -0.827      0.408    -8039.816    3269.358
ar.L1            -0.2364      0.261     -0.906      0.365     -0.748      0.275
ma.L1            0.2863      0.339      0.844      0.399     -0.379      0.951
ar.S.L4            -0.4149      0.107     -3.874      0.000     -0.625     -0.205
ma.S.L4            0.4969      0.190      2.614      0.009      0.124      0.869
sigma2           7.689e+07    16.229    4.74e+06      0.000    7.69e+07    7.69e+07
=====
Ljung-Box (L1) (Q):      0.01    Jarque-Bera (JB):            20.14
Prob(Q):            0.93    Prob(JB):            0.00
Heteroskedasticity (H):    0.21    Skew:            -0.66
Prob(H) (two-sided):    0.00    Kurtosis:            5.29
=====

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 8.69e+22. Standard errors may be unstable.

```

Paso 4. Predicción sobre el conjunto de test

```

In [53]: # Realizamos la predicción sobre los trimestres de test
pred_test = resultado_sarimax.predict(start=y_test.index[0],
                                         end=y_test.index[-1],
                                         exog=exog_test)

# Visualizamos la comparación
plt.figure(figsize=(12,5))
plt.plot(y_train, label="Entrenamiento")
plt.plot(y_test, label="Real (Test)", color='blue')
plt.plot(pred_test, label="Predicción SARIMAX", color='red', linestyle='--')
plt.title("Predicción con SARIMAX (con variables exógenas)")
plt.legend()
plt.grid(True)
plt.show()

```

Predicción con SARIMAX (con variables exógenas)



Paso 5. Evaluación del modelo

```
In [54]: rmse = np.sqrt(mean_squared_error(y_test, pred_test))
mae = mean_absolute_error(y_test, pred_test)

print(f"RMSE (Test): {rmse:.2f}")
print(f"MAE (Test): {mae:.2f}")

RMSE (Test): 6439.55
MAE (Test): 4579.88
```

```
In [55]: comparacion_sarimax = pd.DataFrame({
    "Hipotecas Reales": y_test.values,
    "Predicción SARIMAX": pred_test,
    "Error Absoluto": np.abs(y_test.values - pred_test)
}, index=y_test.index)

print(comparacion_sarimax.round(0))
```

Fecha	Hipotecas Reales	Predicción SARIMAX	Error Absoluto
2023-03-31	109020	110102.0	1082.0
2023-06-30	93695	93618.0	77.0
2023-09-30	88378	82750.0	5628.0
2023-12-31	89873	78339.0	11534.0

Se ha mejorado mucho al incluir las variables. Hay que hacer apartado de conclusiones

4. Comparación en gráficas

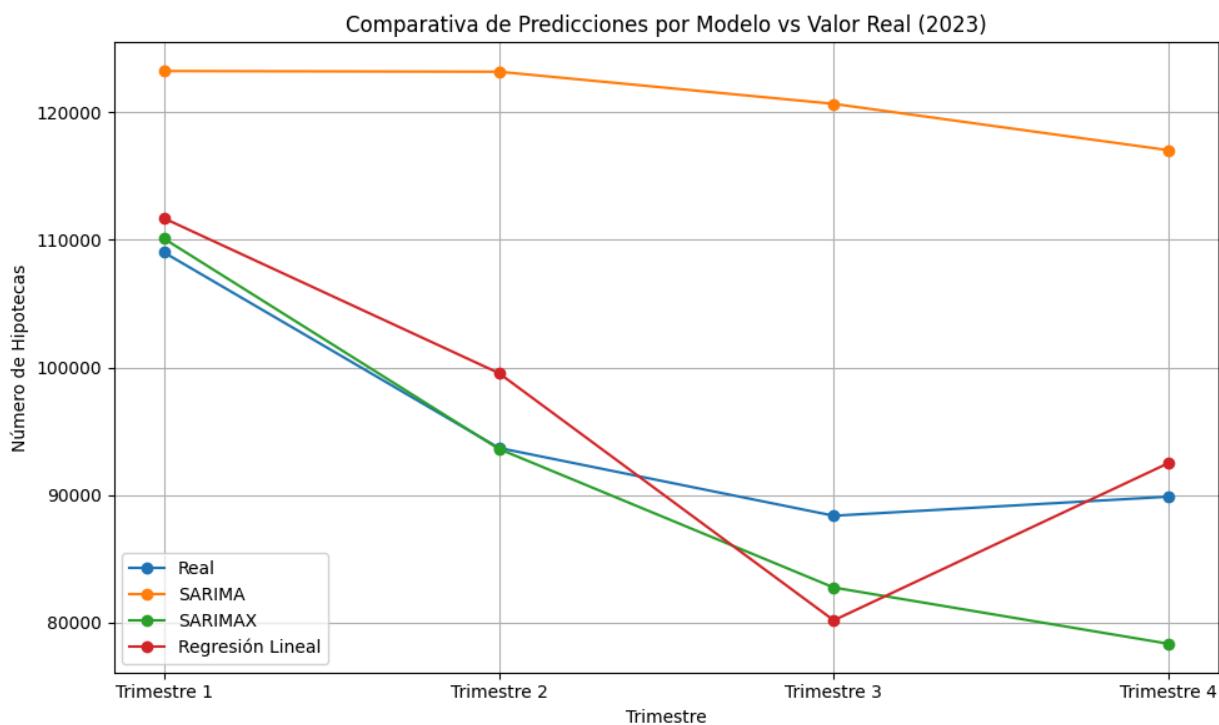
```
In [57]: df_predicciones = pd.DataFrame({
    "Real": y_test,
    "SARIMA": pred_2023_sarima,
    "SARIMAX": pred_test,
    "Regresión Lineal": y_pred_2023
})

df_plot = df_predicciones.reset_index(drop=True)

etiquetas_trimestres = [f"Trimestre {i+1}" for i in range(len(df_plot))]

plt.figure(figsize=(10, 6))
for col in df_plot.columns:
    plt.plot(etiquetas_trimestres, df_plot[col], marker='o', label=col)

plt.title("Comparativa de Predicciones por Modelo vs Valor Real (2023)")
plt.xlabel("Trimestre")
plt.ylabel("Número de Hipotecas")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



In []: