

Market Intelligence Assessment.

- Realizado por Esteban Ramirez

Objetivo

Utilizando la base de datos adjunta (csv) con información de ventas mensuales de hipotecas para el sistema bancario mexicano, el objetivo como Consultor de Market Intelligence es procesar esta base para encontrar insights relevantes y oportunidades de mercado para Scotiabank.

La base de datos contiene información agregada a nivel mensual sobre las hipotecas otorgadas por diferentes bancos en México. Cada fila representa el monto promedio de crédito, junto con otras características, para un grupo de personas que comparten las mismas variables categóricas (estado, tipo de acreditado, género, etc.).

Estructura de la Base de Datos:

- Variables Categóricas (7):
 - cve_periodo: Fecha (año y mes)
 - nombre_publicacion: Nombre del banco
 - dl_estado: Estado
 - tipo_acreditado: Público o privado
 - dl_tipo_comprobacion: Asalariado o no asalariado
 - dl_genero: Género
 - dl_destino_credito: Destino del crédito
- Variables Numéricas (5):
 - dat_ingreso: Ingreso mensual bruto promedio
 - dat_valor_vivienda_originacion: Valor de la vivienda
 - dat_ai_edad_acred: Edad del acreditado
 - dat_monto_orig_cred: Monto del crédito
 - tasa ponderada: Tasa de interés ponderada

Limpieza y transformacion de datos

Importando datos y eliminando acentos de variables categoricas

```
In [7]: import pandas as pd
import unicodedata

# Cargar el archivo CSV
df = pd.read_csv("Hipotecarios_Marginales febrero2024conBBVA.csv", encoding="latin1")
```

```

# Definir las columnas categóricas
columnas_categoricas = [
    "cve_periodo",
    "nombre_publicacion",
    "dl_estado",
    "tipo_acreditado",
    "dl_tipo_comprobacion",
    "dl_genero",
    "dl_destino_credito"
]

# Función para eliminar acentos
def quitar_acentos(texto):
    if isinstance(texto, str):
        texto_normalizado = unicodedata.normalize("NFKD", texto)
        return "".join([c for c in texto_normalizado if not unicodedata.combining(c)])
    return texto

# Aplicar la función a las columnas categóricas
for col in columnas_categoricas:
    df[col] = df[col].apply(quitar_acentos)

# Verificar los resultados
print(df[columnas_categoricas].head())

```

	cve_periodo	nombre_publicacion	dl_estado \
0	201901	Banjercito	SAN LUIS POTOSI
1	201901	Banjercito	VERACRUZ DE IGNACIO DE LA LLAVE
2	201901	Banjercito	YUCATAN
3	201901	Banobras	OAXACA
4	201902	Banjercito	CIUDAD DE MEXICO

	tipo_acreditado	dl_tipo_comprobacion	dl_genero \
0	Asalariado	Publico	Asalariado Masculino
1	Asalariado	Publico	Asalariado Masculino
2	Asalariado	Publico	Asalariado Masculino
3	Asalariado	Publico	Asalariado Femenino
4	Asalariado	Publico	Asalariado Masculino

	dl_destino_credito
0	Adquisicion de Vivienda Usada
1	Adquisicion de Vivienda Nueva
2	Adquisicion de Vivienda Usada
3	Adquisicion de Vivienda Usada
4	Adquisicion de Vivienda Usada

Seleccionando estructura de la base de datos

```

In [8]: # Definir las columnas para conservar
columnas_deseadas = [
    "cve_periodo",
    "nombre_publicacion",
    "dl_estado",
    "tipo_acreditado",
    "dl_tipo_comprobacion",

```

```
"dl_genero",
"dl_destino_credito",
"dat_ingreso_mensual_bruto",
"dat_valor_vivienda_originacion",
"dat_ai_edad_acred",
"dat_monto_orig_cred",
"tasa_ponderada"
]

# Filtrar el DataFrame para conservar solo esas columnas
df_filtrado = df[columnas_deseadas]

# Guardar el nuevo archivo Limpio
# df_filtrado.to_csv("hipotecas_limpio.csv", index=False, encoding="utf-8")

# Mostrar las primeras filas para verificar
#print(df_filtrado.head())
```

In [9]: # Agrupar por todas las variables categóricas
grupo = df.groupby(columnas_categoricas, as_index=False)

Calculando tasas ponderadas

In [10]: # Crear columna de ingreso por intereses
df["ingreso_intereses"] = df["tasa_ponderada"] * df["dat_monto_orig_cred"]

Agrupar y calcular tasa ponderada agrupada
resultado = df.groupby(columnas_categoricas).agg({
 "ingreso_intereses": "sum",
 "dat_monto_orig_cred": "sum"
}).reset_index()

Calcular tasa ponderada agrupada
resultado["tasa_ponderada_agrupada"] = resultado["ingreso_intereses"] / resultado["dat_monto_orig_cred"]

Mostrar resultados
print(resultado[columnas_categoricas + ["tasa_ponderada_agrupada"]].head())

```

      cve_período nombre_publicacion      dl_estado      tipo_acreditado \
0        201901             Afirme  AGUASCALIENTES      No Asalariado
1        201901             Afirme  BAJA CALIFORNIA  Asalariado Privado
2        201901             Afirme  BAJA CALIFORNIA      No Asalariado
3        201901             Afirme CIUDAD DE MEXICO  Asalariado Privado
4        201901             Afirme CIUDAD DE MEXICO      No Asalariado

      dl_tipo_comprobacion  dl_genero      dl_destino_credito \
0       No asalariado    Masculino  Adquisicion de Vivienda Usada
1     Asalariado          Masculino  Adquisicion de Vivienda Usada
2       No asalariado    Femenino   Adquisicion de Vivienda Usada
3     Asalariado          Femenino   Adquisicion de Vivienda Usada
4       No asalariado    Masculino  Adquisicion de Vivienda Usada

      tasa_ponderada_agrupada
0            12.500000
1            11.524682
2            12.500000
3            10.990000
4            10.990000

```

1. Participación de mercado de Scotiabank (Nov'23)

```
In [11]: # Filtrar por Scotiabank y noviembre de 2023.
scotiabank_nov2023 = df[(df["nombre_publicacion"] == "Scotiabank") & (df["cve_período"] == 202311)]

# Crédito total originado por Scotiabank
scotiabank_total = scotiabank_nov2023["dat_monto_orig_cred"].sum()

# Crédito total originado a nivel nacional en noviembre de 2023
national_nov2023 = df[df["cve_período"] == 202311]
national_total = national_nov2023["dat_monto_orig_cred"].sum()

# Cálculo de la participación de mercado
participacion = (scotiabank_total / national_total) * 100 if national_total > 0 else 0
print(f"Participación de Scotiabank en noviembre 2023: {participacion:.2f}%")
```

Participación de Scotiabank en noviembre 2023: 21.02%

2. Evolución de la participación de mercado

```
In [12]: # Analizando evolución de participación de mercado de Scotiabank en hipotecas

import unicodedata
import matplotlib.pyplot as plt

# Agrupar por mes y calcular montos
scotiabank_mensual = df[df["nombre_publicacion"].str.upper() == "SCOTIABANK"].groupby("cve_período").sum()

# Calcular participación de mercado
participacion_mensual = scotiabank_mensual["dat_monto_orig_cred"] / scotiabank_mensual["dat_monto_orig_cred"].sum()
```

```

participacion = (scotiabank_mensual / total_mensual).fillna(0) * 100
participacion = participacion.reset_index()
participacion.columns = ["cve_periodo", "participacion_scotiabank"]

print(participacion.head())

    cve_periodo  participacion_scotiabank
0      201901           12.577624
1      201902           15.001368
2      201903           11.024280
3      201904           11.587323
4      201905           11.697966

```

Ajustando detalles de gráfica para mejorar visualización

```

In [24]: import pandas as pd
import matplotlib.pyplot as plt

# Convertir cve_periodo a formato de fecha
participacion["fecha"] = pd.to_datetime(participacion["cve_periodo"].astype(str), f

# Manual mapping for Spanish month abbreviations
meses_espanol = {
    1: 'ene', 2: 'feb', 3: 'mar', 4: 'abr', 5: 'may', 6: 'jun',
    7: 'jul', 8: 'ago', 9: 'sep', 10: 'oct', 11: 'nov', 12: 'dic'
}

# Format the date using the manual mapping
participacion["etiqueta_mes"] = participacion["fecha"].dt.month.map(meses_espanol)

# Get the last month's participation for annotation
last_month_label = participacion["etiqueta_mes"].iloc[-1]
last_month_participation = participacion["participacion_scotiabank"].iloc[-1]

plt.style.use("seaborn-v0_8-whitegrid")
fig, ax = plt.subplots(figsize=(14, 7))

# Plotting with executive style
ax.plot(participacion["etiqueta_mes"], participacion["participacion_scotiabank"], m

# Adding title and labels
ax.set_title("Evolución de la Participación de Scotiabank en Originación de Hipotecas", fontweight="bold", fontsize=14)
ax.set_xlabel("Mes", fontsize=12)
ax.set_ylabel("Participación de Mercado (%)", fontsize=12)

# Improve x-axis labels and rotation
ax.set_xticks(participacion["etiqueta_mes"][::6]) # Show every 6th month Label
ax.tick_params(axis="x", rotation=45)

# Adding grid lines
ax.grid(axis="y", linestyle="--", alpha=0.7)

# Adding annotation for the last data point
ax.annotate(f'{last_month_participation:.2f}%', xy=(12, last_month_participation), xytext=(-10, -10), textcoords="offset pixels", arrowprops={"arrowheadcolor": "black", "arrowstyle": "solid", "color": "black", "shape": "triangle-down", "width": 2}, fontweight="bold", fontstyle="italic", fontsize=12)

```

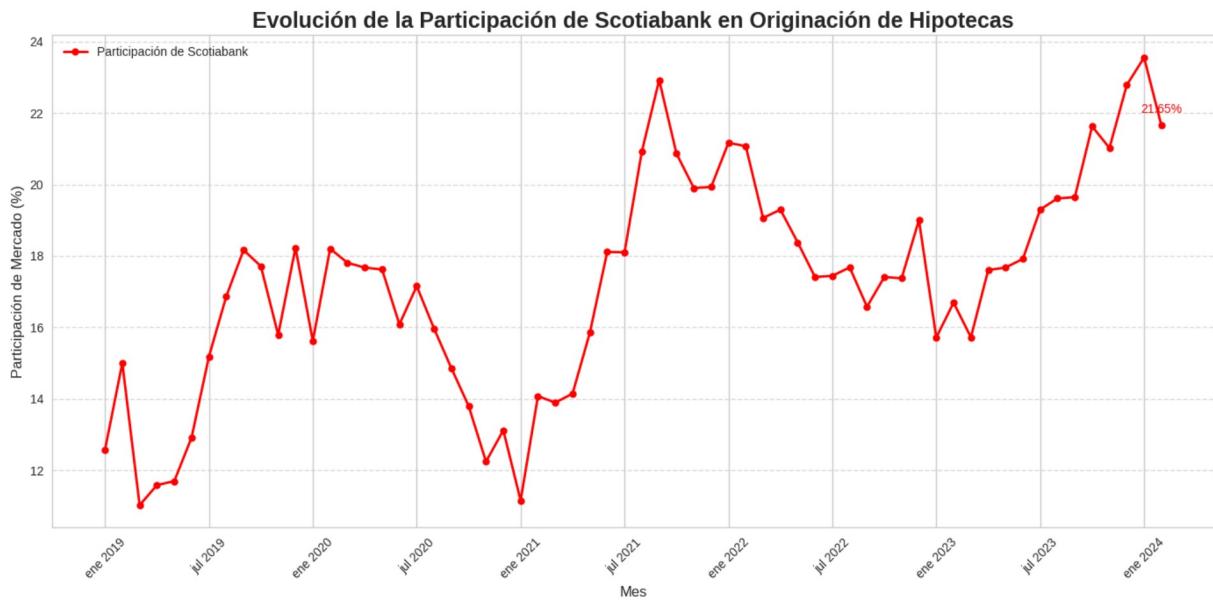
```

        (last_month_label, last_month_participation),
        textcoords="offset points",
        xytext=(0,10),
        ha='center',
        fontsize=10,
        color="red")

# Adding legend
ax.legend(loc="upper left")

# Adjust layout and show plot
fig.tight_layout()
plt.show()

```



3. Top 5 participantes por participación de mercado:

```
In [14]: # Calcular monto total originado por cada banco
monto_por_banco = df.groupby("nombre_publicacion")["dat_monto_orig_cred"].sum().res

# Calcular monto total nacional
monto_total_nacional = monto_por_banco["dat_monto_orig_cred"].sum()

# Calcular participación de mercado
monto_por_banco["participacion_mercado"] = monto_por_banco["dat_monto_orig_cred"] / monto_total_nacional

# Ordenar de mayor a menor
top_bancos = monto_por_banco.sort_values(by="participacion_mercado", ascending=False)

# Mostrar resultados
print(top_bancos[["nombre_publicacion", "participacion_mercado"]])
```

```

    nombre_publicacion  participacion_mercado
2           BBVA Mexico          22.810123
12          Banorte            18.853594
28         Scotiabank          17.480181
26        Santander           14.499790
16           HSBC              12.745977

```

In [22]: # Presentar La tabla con formato ejecutivo

```

print("Top 5 Participantes por Participación de Mercado:")
print(top_bancos[["nombre_publicacion", "participacion_mercado"]].to_string(index=False))

```

Top 5 Participantes por Participación de Mercado:

nombre_publicacion	participacion_mercado
BBVA Mexico	22.81%
Banorte	18.85%
Scotiabank	17.48%
Santander	14.50%
HSBC	12.75%

4. Concentración geográfica:

In [15]: # Identificando el principal competidor (mayor monto total originado)

```

monto_por_banco = df.groupby("nombre_publicacion")["dat_monto_orig_cred"].sum()
principal_competidor = monto_por_banco.drop("Scotiabank", errors="ignore").idxmax()
#Agrupando por estado y banco
grupo_estado_banco = df.groupby(["dl_estado", "nombre_publicacion"])["dat_monto_ori

```

Separando datos de Scotiabank y del principal competidor

```

scotia = grupo_estado_banco[grupo_estado_banco["nombre_publicacion"] == "Scotiabank"]
competidor = grupo_estado_banco[grupo_estado_banco["nombre_publicacion"] == principal_competidor]

```

Unir ambos por estado

```

comparacion = pd.merge(scotia, competidor, on="dl_estado", suffixes=("_scotia", "_comp

```

Calcular ratio de originación

```

comparacion["ratio_scotia_vs_competidor"] = comparacion["dat_monto_orig_cred_scotia"] / c

```

Seleccionar Los 5 estados donde Scotiabank supera al competidor

```

top_estados = comparacion[comparacion["ratio_scotia_vs_competidor"] > 1].sort_values(
    by="ratio_scotia_vs_competidor", ascending=False).head(5)

```

Mostrar resultados

```

print("Principal competidor:", principal_competidor)
print("\nTop 5 estados donde Scotiabank supera al competidor:\n")
print(top_estados[["dl_estado", "dat_monto_orig_cred_scotia", "dat_monto_orig_cred_c

```

Principal competidor: BBVA Mexico

Top 5 estados donde Scotiabank supera al competidor:

	dl_estado	dat_monto_orig_cred_scotia \
28	TLAXCALA	3.573571e+08
7	COAHUILA DE ZARAGOZA	8.075784e+09
29	VERACRUZ DE IGNACIO DE LA LLAVE	6.063185e+09
6	CIUDAD DE MEXICO	4.405078e+10
30	YUCATAN	7.559042e+09

	dat_monto_orig_cred_competidor	ratio_scotia_vs_competidor
28	2.954934e+08	1.209357
7	6.846366e+09	1.179572
29	5.219963e+09	1.161538
6	3.960474e+10	1.112260
30	7.408986e+09	1.020253

In [23]: *# Presentar la tabla con formato ejecutivo*

```
print("Top 5 estados donde Scotiabank supera a", principal_competidor, ":")  
print(top_estados[["dl_estado", "dat_monto_orig_cred_scotia", "dat_monto_orig_cred_
```

Top 5 estados donde Scotiabank supera a BBVA Mexico :

	dl_estado	dat_monto_orig_cred_scotia	dat_monto_orig_cred_compe tidor	ratio_scotia_vs_competidor
	TLAXCALA	357,357,129		295,49
3,432	1.21			
	COAHUILA DE ZARAGOZA	8,075,784,145		6,846,36
5,728	1.18			
	VERACRUZ DE IGNACIO DE LA LLAVE	6,063,184,761		5,219,96
3,164	1.16			
	CIUDAD DE MEXICO	44,050,778,914		39,604,74
4,249	1.11			
	YUCATAN	7,559,042,115		7,408,98
5,534	1.02			

5. Identificar Oportunidades:

Identificando segmentos de clientes potenciales desatendidos por Scotiabank

In [16]: *# Filtrar datos de Scotiabank*

```
scotia_df = df[df["nombre_publicacion"].str.upper() == "SCOTIABANK"]  
  
# Creando rangos de edad  
def clasificar_edad(edad):  
    if edad < 30:  
        return "<30"  
    elif edad < 40:  
        return "30-39"  
    elif edad < 50:  
        return "40-49"  
    elif edad < 60:  
        return "50-59"  
    else:
```

```

        return "60+"

df["rango_edad"] = df["dat_ai_edad_acred"].apply(clasificar_edad)
scotia_df["rango_edad"] = scotia_df["dat_ai_edad_acred"].apply(clasificar_edad)

# Creando rangos de ingreso
def clasificar_ingreso(ingreso):
    if ingreso < 10000:
        return "<10k"
    elif ingreso < 20000:
        return "10k-20k"
    elif ingreso < 30000:
        return "20k-30k"
    else:
        return "30k+"

df["rango_ingreso"] = df["dat_ingreso_mensual_bruto"].apply(clasificar_ingreso)
scotia_df["rango_ingreso"] = scotia_df["dat_ingreso_mensual_bruto"].apply(clasificar_ingreso)

# Agrupando y calculando metricas
def calcular_participacion(df_total, df_banco, columna_segmento):
    total = df_total.groupby(columna_segmento).agg({
        "dat_monto_orig_cred": "sum",
        columna_segmento: "count"
    }).rename(columns={"dat_monto_orig_cred": "monto_total", columna_segmento: "registro_scotia"})

    banco = df_banco.groupby(columna_segmento).agg({
        "dat_monto_orig_cred": "sum",
        columna_segmento: "count"
    }).rename(columns={"dat_monto_orig_cred": "monto_scotia", columna_segmento: "registro_scotia"})

    comparacion = total.join(banco, how="left").fillna(0)
    comparacion["participacion_monto"] = comparacion["monto_scotia"] / comparacion["monto_total"]
    comparacion["participacion_registros"] = comparacion["registros_scotia"] / comparacion["registro_scotia"]
    return comparacion

# Partcipacion edad
edad_part = calcular_participacion(df, scotia_df, "rango_edad")

# Participacion ingreso
ingreso_part = calcular_participacion(df, scotia_df, "rango_ingreso")

# Participacion por estado
estado_part = calcular_participacion(df, scotia_df, "dl_estado")

```

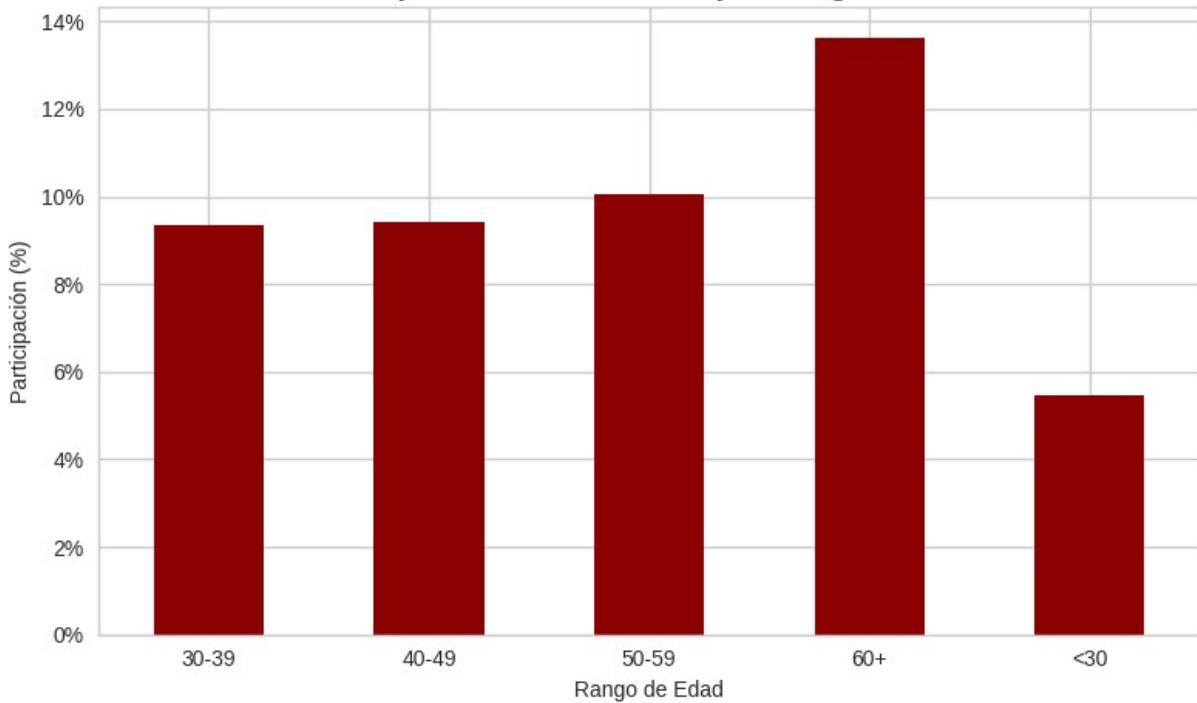
```
/tmp/ipython-input-1399641226.py:18: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy  
    scotia_df["rango_edad"] = scotia_df["dat_ai_edad_acred"].apply(clasificar_edad)  
/tmp/ipython-input-1399641226.py:32: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy  
    scotia_df["rango_ingreso"] = scotia_df["dat_ingreso_mensual_bruto"].apply(clasificar_ingreso)
```

Generando visualizaciones

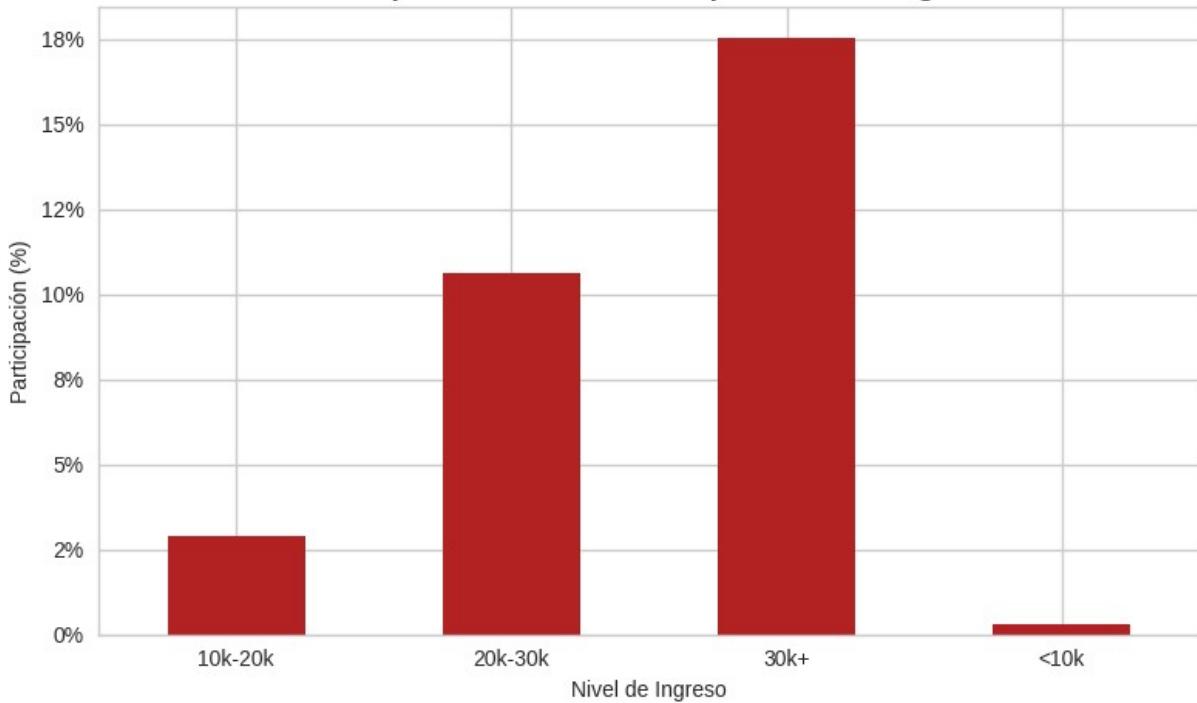
```
In [28]: import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd # Import pandas as it's used for DataFrame  
  
# Gráfica Partcipacion edad  
plt.style.use("seaborn-v0_8-whitegrid") # Using a professional style  
fig1, ax1 = plt.subplots(figsize=(8, 5))  
edad_part["participacion_registros"].plot(kind="bar", ax=ax1, color="darkred", label="Participación (%)")  
ax1.set_title("Participación de Scotiabank por Rango de Edad", fontsize=14, fontweight="bold")  
ax1.set_ylabel("Participación (%)", fontsize=10)  
ax1.set_xlabel("Rango de Edad", fontsize=10) # Added xlabel  
ax1.set_xticklabels(edad_part.index, rotation=0)  
ax1.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: '{:.0%}'.format(x))) #  
fig1.tight_layout()  
  
# Grafica Partcipacion ingreso  
plt.style.use("seaborn-v0_8-whitegrid") # Using a professional style  
fig2, ax2 = plt.subplots(figsize=(8, 5))  
ingreso_part["participacion_registros"].plot(kind="bar", ax=ax2, color="firebrick", label="Participación (%)")  
ax2.set_title("Participación de Scotiabank por Nivel de Ingreso", fontsize=14, fontweight="bold")  
ax2.set_ylabel("Participación (%)", fontsize=10)  
ax2.set_xlabel("Nivel de Ingreso", fontsize=10) # Added xlabel  
ax2.set_xticklabels(ingreso_part.index, rotation=0)  
ax2.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: '{:.0%}'.format(x))) #  
fig2.tight_layout()  
  
# Grafica Mapa de calor por estado  
plt.style.use("seaborn-v0_8-whitegrid") # Using a professional style  
fig3, ax3 = plt.subplots(figsize=(10, 8))  
estado_part_sorted = estado_part.sort_values("participacion_registros", ascending=False)  
heatmap_data = estado_part_sorted["participacion_registros"].values.reshape(-1, 1)  
im = ax3.imshow(heatmap_data, cmap="Reds", aspect="auto") # Changed colormap to Red  
ax3.set_yticks(np.arange(len(estado_part_sorted)))  
ax3.set_yticklabels(estado_part_sorted.index)  
ax3.set_xticks([]) # Removed x-axis ticks  
ax3.set_title("Mapa de Calor: Participación de Scotiabank por Estado", fontsize=14)  
fig3.colorbar(im, ax=ax3, format=plt.FuncFormatter(lambda x, _: '{:.0%}'.format(x)))
```

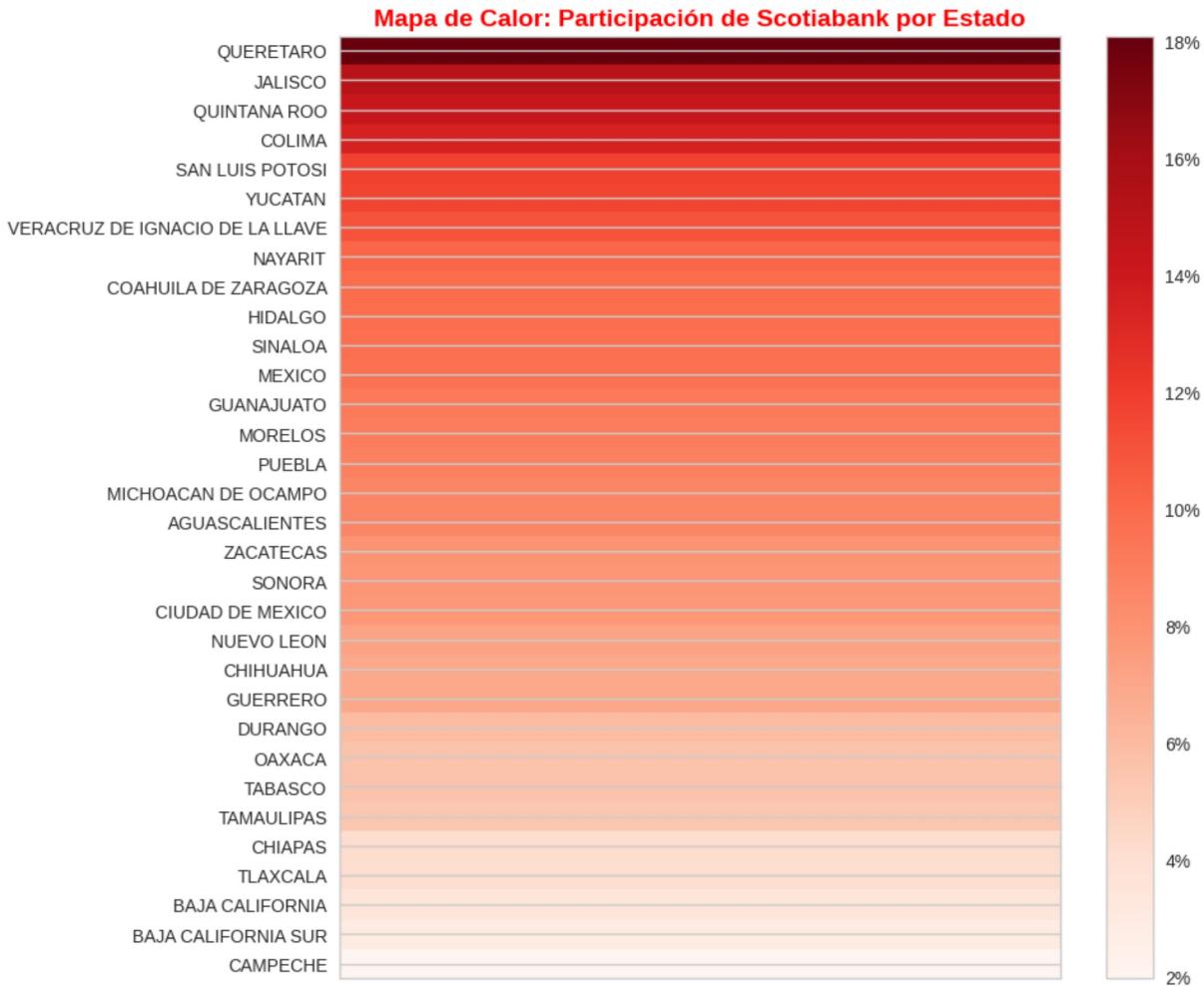
```
fig3.tight_layout()
```

Participación de Scotiabank por Rango de Edad



Participación de Scotiabank por Nivel de Ingreso





In []:

Explorando regiones geográficas con tasas de interés promedio más altas donde Scotiabank podría ajustar su tasa de interés.

```
In [38]: # Filtrar datos donde el banco NO sea Scotiabank
df_filtrado = df[df["nombre_publicacion"].str.lower() != "scotiabank"]

# Agrupar por estado y calcular tasa promedio
tasas_por_estado = df_filtrado.groupby("dl_estado")["tasa_ponderada"].mean().reset_
tasas_por_estado.columns = ["Estado", "Tasa_Promedio"]
# Ordenar de mayor a menor
tasas_por_estado = tasas_por_estado.sort_values(by="Tasa_Promedio", ascending=False

# Calcular promedio nacional y filtrar estados con tasas significativamente más alt
promedio_nacional = tasas_por_estado["Tasa_Promedio"].mean()
umbral = promedio_nacional + 1.5 # Ajuste arbitrario para definir "significativamente más alt"
estados_altas_tasas = tasas_por_estado[tasas_por_estado["Tasa_Promedio"] > umbral]

print("Promedio nacional de tasa de interés:", round(promedio_nacional, 2))
print("Estados con tasas significativamente más altas:")
print(estados_altas_tasas)
```

Promedio nacional de tasa de interés: 12.72
 Estados con tasas significativamente más altas:

	Estado	Tasa_Promedio
28	TLAXCALA	14.885732
3	CAMPECHE	14.500633
6	CIUDAD DE MEXICO	14.378687
9	DURANGO	14.298374

```
In [36]: # Generando visualización ejecutiva de tasas de interés promedio por estado
import matplotlib.pyplot as plt

# Datos
estados = ["TLAXCALA", "CAMPECHE", "CIUDAD DE MEXICO", "DURANGO"]
tasas = [14.89, 14.50, 14.38, 14.30]
promedio_nacional = 12.72

# Estilo ejecutivo tipo Power BI
plt.style.use("seaborn-v0_8")
fig, ax = plt.subplots(figsize=(12, 6))

# Paleta de colores rojos institucionales
colores = ["#B22222", "#CD5C5C", "#8B0000", "#A52A2A"]

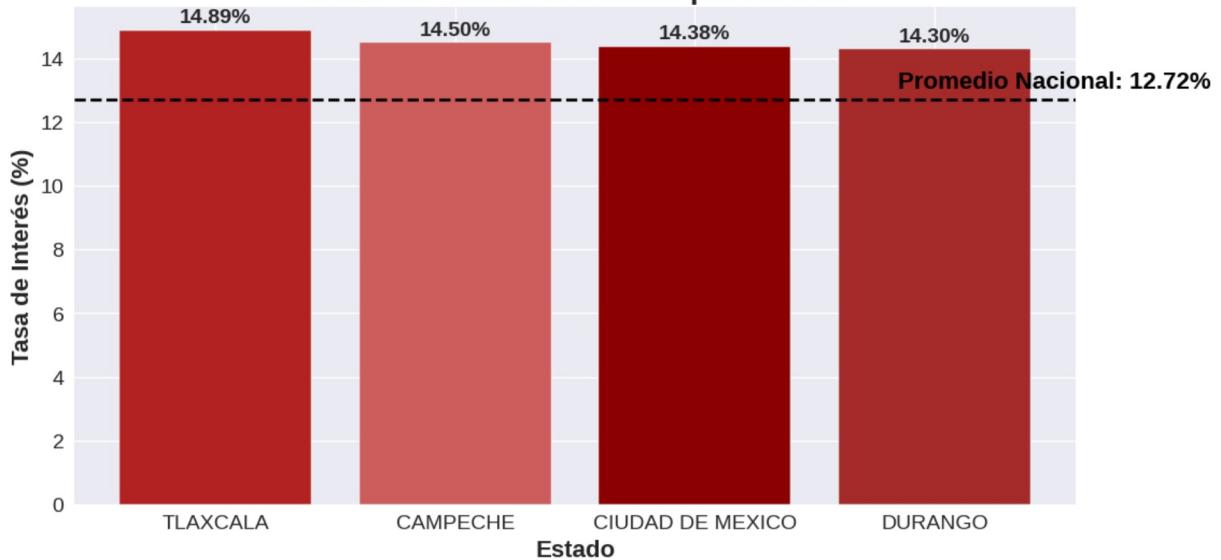
# Crear barras
barras = ax.bar(estados, tasas, color=colores)

# Etiquetas sobre cada barra
for barra in barras:
    altura = barra.get_height()
    ax.text(barra.get_x() + barra.get_width() / 2, altura + 0.1,
            f"{altura:.2f}%", ha='center', va='bottom', fontsize=14, fontweight='bold')

# Línea de referencia del promedio nacional
ax.axhline(promedio_nacional, color="black", linestyle="--", linewidth=2)
ax.text(len(estados) - 0.5, promedio_nacional + 0.2,
        f"Promedio Nacional: {promedio_nacional:.2f}%", ha='center', va='bottom',
        fontsize=16, fontweight='bold', color='black')

# Títulos y etiquetas
ax.set_title("Tasas de Interés Promedio por Estado", fontsize=20, fontweight='bold')
ax.set_ylabel("Tasa de Interés (%)", fontsize=16, fontweight='bold')
ax.set_xlabel("Estado", fontsize=16, fontweight='bold')
ax.tick_params(axis='x', labelsize=14)
ax.tick_params(axis='y', labelsize=14)
```

Tasas de Interés Promedio por Estado



```
In [ ]: !jupyter nbconvert --to html "Market_Intelligence_Assessment_Sciabank.ipynb"
from google.colab import files
files.download("Market_Intelligence_Assessment_Sciabank.html")
```