In [15]:
```
import pandas as pd

# Cargar el archivo CSV
df = pd.read_csv('/content/drive/MyDrive/DATATHON/Venta.csv')

# Verificar que las columnas esperadas existan
expected_columns = ['TIENDA_ID', 'MES_ID', 'VENTA_TOTAL']
if not all(col in df.columns for col in expected_columns):
    print("Error: El archivo CSV debe contener las columnas 'TIENDA_ID', 'MES_ID' y
else:
    # Calcular promedio de VENTA_TOTAL por tienda, usando todos sus registros actua
    resumen = df.groupby('TIENDA_ID')['VENTA_TOTAL'].mean().reset_index()
    resumen.columns = ['TIENDA_ID', 'PROMEDIO_VENTA_TOTAL']

    # Guardar el resumen en un archivo CSV
    output_path = '/content/drive/MyDrive/DATATHON/Promedio_Venta.csv'
    resumen.to_csv(output_path, index=False)
    print(f"\nArchivo exportado exitosamente a: {output_path}")

    # Imprimir el resumen
    print("\nResumen de tiendas con su promedio de venta:")
    print(resumen)

# Cargar los archivos
promedio_venta_path = '/content/drive/MyDrive/DATATHON/Promedio_Venta.csv'
dim_tienda_path = '/content/drive/MyDrive/DATATHON/DIM_TIENDA_TEST.csv'

promedio_df = pd.read_csv(promedio_venta_path)
dim_tienda_df = pd.read_csv(dim_tienda_path)

# Verificar que la columna TIENDA_ID exista en ambos DataFrames
if 'TIENDA_ID' not in promedio_df.columns or 'TIENDA_ID' not in dim_tienda_df.colum
    print("Error: Ambos archivos deben contener la columna 'TIENDA_ID'")
else:
    # Realizar el merge
    merged_df = dim_tienda_df.merge(promedio_df, on='TIENDA_ID', how='left')

    # Guardar el archivo combinado
    output_path = '/content/drive/MyDrive/DATATHON/DIM_TIENDA_TEST_Con_Promedio.csv
    merged_df.to_csv(output_path, index=False)
    print(f"\nArchivo combinado exportado exitosamente a: {output_path}")

    # Mostrar los primeros registros del archivo combinado
    print("\nPrimeros registros del archivo combinado:")
    print(merged_df.head())

dim_tienda_path = '/content/drive/MyDrive/DATATHON/DIM_TIENDA.csv'
dim_tienda_train_df = pd.read_csv(dim_tienda_path)

# Verificar que la columna TIENDA_ID exista en ambos DataFrames
if 'TIENDA_ID' not in promedio_df.columns or 'TIENDA_ID' not in dim_tienda_train_df
    print("Error: Ambos archivos deben contener la columna 'TIENDA_ID'")
else:
    # Realizar el merge
```

```
merged_df_train = dim_tienda_train_df.merge(promedio_df, on='TIENDA_ID', how='l

# Guardar el archivo combinado
output_path = '/content/drive/MyDrive/DATATHON/DIM_TIENDA_TRAIN_Con_Promedio.cs
merged_df_train.to_csv(output_path, index=False)
print(f"\nArchivo combinado exportado exitosamente a: {output_path}")

# Mostrar los primeros registros del archivo combinado
print("\nPrimeros registros del archivo combinado:")
print(merged_df_train.head())
```

```
Archivo exportado exitosamente a: /content/drive/MyDrive/DATATHON/Promedio_Venta.csv

Resumen de tiendas con su promedio de venta:
      TIENDA_ID  PROMEDIO_VENTA_TOTAL
0             1          7.080276e+05
1             2          8.994741e+05
2             3          8.274871e+05
3             4          1.247370e+06
4             5          1.596267e+06
...         ...                   ...
1048       1052          1.136733e+06
1049       1053          7.660571e+05
1050       1054          1.561485e+06
1051       1055          1.014392e+06
1052       1056          1.284810e+06

[1053 rows x 2 columns]

Archivo combinado exportado exitosamente a: /content/drive/MyDrive/DATATHON/DIM_TIEN
DA_TEST_Con_Promedio.csv

Primeros registros del archivo combinado:
   TIENDA_ID  PLAZA_CVE NIVELSOCIOECONOMICO_DES ENTORNO_DES  MTS2VENTAS_NUM  \
0        680          1                      C       Hogar          102.36
1        730          1                      C       Hogar           97.43
2        650          1                      C       Hogar          117.01
3        670          1                      C        Base          109.76
4        800          1                      C    Peatonal            0.00

   PUERTASREFRIG_NUM  CAJONESESTACIONAMIENTO_NUM  LATITUD_NUM  LONGITUD_NUM  \
0                 13                           0     25.65488    -100.21207
1                 14                           0     25.66315    -100.22738
2                 13                           0     25.66404    -100.22993
3                 13                           0     25.66508    -100.26338
4                  0                           0     25.69367    -100.21433

  SEGMENTO_MAESTRO_DESC LID_UBICACION_TIENDA DATASET  PROMEDIO_VENTA_TOTAL
0         Hogar Reunión          UT_DENSIDAD    TEST          6.503506e+05
1         Hogar Reunión          UT_DENSIDAD    TEST          1.324069e+06
2         Hogar Reunión          UT_DENSIDAD    TEST          9.372804e+05
3         Hogar Reunión          UT_DENSIDAD    TEST          7.728227e+05
4        Parada Técnica  UT_TRAFICO_PEATONAL    TEST          5.228823e+05

Archivo combinado exportado exitosamente a: /content/drive/MyDrive/DATATHON/DIM_TIEN
DA_TRAIN_Con_Promedio.csv

Primeros registros del archivo combinado:
   TIENDA_ID  PLAZA_CVE NIVELSOCIOECONOMICO_DES ENTORNO_DES  MTS2VENTAS_NUM  \
0        126          1                     BC       Hogar          127.42
1        681          1                      C       Hogar          128.13
2        117          1                      C        Base           87.62
3        682          1                      C       Hogar           90.70
4        275          1                      C       Hogar           95.30

   PUERTASREFRIG_NUM  CAJONESESTACIONAMIENTO_NUM  LATITUD_NUM  LONGITUD_NUM  \
0                 13                           7     25.69107    -100.21261
```

```
1                 13                          0     25.73571    -100.18086
2                 11                         11     25.71883    -100.19133
3                 13                          0     25.66952    -100.20744
4                 13                          6     25.73766    -100.16116

  SEGMENTO_MAESTRO_DESC LID_UBICACION_TIENDA DATASET  PROMEDIO_VENTA_TOTAL
0         Hogar Reunión          UT_DENSIDAD   TRAIN         897943.461429
1         Hogar Reunión          UT_DENSIDAD   TRAIN         875823.110000
2         Hogar Reunión          UT_DENSIDAD   TRAIN         454438.265714
3         Hogar Reunión          UT_DENSIDAD   TRAIN         945847.573810
4         Hogar Reunión          UT_DENSIDAD   TRAIN         859411.378095
```

Carga de archivo con el promedio de ventas por tienda

In [16]:
```
from google.colab import drive
drive.mount('/content/drive')


import pandas as pd
df1=pd.read_csv("/content/drive/MyDrive/DATATHON/DIM_TIENDA_TRAIN_Con_Promedio.csv"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
mount("/content/drive", force_remount=True).
```

1. Imputación de datos en MTS2VENTAS_NUM con el promedio de todos los datos

In [17]:
```
import numpy as np

promedio=np.mean(df1['MTS2VENTAS_NUM'])
print(promedio)

for valor in df1['MTS2VENTAS_NUM']:
  if valor==0:
    df1['MTS2VENTAS_NUM'].replace(0, promedio, inplace=True)
```

```
78.8974553101998
<ipython-input-17-51e78530e3a8>:8: FutureWarning: A value is trying to be set on a c
opy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df1['MTS2VENTAS_NUM'].replace(0, promedio, inplace=True)
```

2. Eliminación de variable CAJONESESTACIONAMIENTO_NUM, debido a que no se sabe si son valores NaN o si en realidad la sucursal tiene 0 cajones de estacionamiento se elimina ya que los ceros conforman 50% de los datos dentro de esta variable

In [18]:
```
df1=df1.drop('CAJONESESTACIONAMIENTO_NUM', axis=1)
```

3. Imputación de datos en PUERTASREFRIG_NUM con el promedio redondeado

In [19]:
```
mean1=np.mean(df1['PUERTASREFRIG_NUM'])
print(mean1)
mean_refrigeradores=round(mean1)
print(mean_refrigeradores)

for valor in df1['PUERTASREFRIG_NUM']:
  if valor==0:
    df1['PUERTASREFRIG_NUM'].replace(0, mean_refrigeradores, inplace=True)
```

```
9.468980021030495
9
```

```
<ipython-input-19-d0b5a5dd4059>:8: FutureWarning: A value is trying to be set on a c
opy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df1['PUERTASREFRIG_NUM'].replace(0, mean_refrigeradores, inplace=True)
```

4. Imputación de valores NaN en la variable SEGMENTO_MAESTRO_DESC con la moda para cada ENTORNO_DES

In [20]:
```
df1['SEGMENTO_MAESTRO_DESC'].replace(['Base', 'Hogar', 'Receso'], 'Hogar Reunion',
```

```
<ipython-input-20-5a98162e3868>:1: FutureWarning: A value is trying to be set on a c
opy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df1['SEGMENTO_MAESTRO_DESC'].replace(['Base', 'Hogar', 'Receso'], 'Hogar Reunion',
inplace=True)
```

5. Eliminación de variable DATASET porque solo es identificación de TRAIN y TEST

In [21]:
```
df1=df1.drop('DATASET', axis=1)
```

6. Descarga de archivo csv

In [22]:
```
df1.to_csv('DIM_TIENDA_TRAIN_IMPUTADO2.csv', index=False, encoding='utf-8')
```

In [23]:
```
df1.to_csv('/content/drive/MyDrive/DATATHON/DIM_TIENDA_TRAIN_IMPUTADO2.csv', index=
```

7. Agrupación por clustering dependiendo de las coordenadas para mantener las ubicaciones como variables categóricas.

Se juntan datos TRAIN y TEST para que haga los mismos clusterings y después se vuelven a separar

In [25]:
```
import folium
from sklearn.cluster import KMeans

data_train = pd.read_csv("/content/drive/MyDrive/DATATHON/DIM_TIENDA_TRAIN_IMPUTADO
data_test = pd.read_csv("/content/drive/MyDrive/DATATHON/DIM_TIENDA_TEST_Con_Promed

data_test=data_test.drop(['CAJONESESTACIONAMIENTO_NUM', 'DATASET'], axis=1)

#Combinamos Train y Test para que los clusters se mantengan iguales para ambos
data_combined = pd.concat([data_train, data_test], ignore_index=True)

coords_combined = data_combined[['LATITUD_NUM', 'LONGITUD_NUM']]

#K-means
k = 6  #Número de clusters
kmeans = KMeans(n_clusters=k, random_state=42).fit(coords_combined)

#Agregamos clusters como variable
data_combined['cluster'] = kmeans.labels_

#Separación train y test
data_train_with_clusters = data_combined.iloc[:len(data_train)].copy()
data_test_with_clusters = data_combined.iloc[len(data_train):].copy()

#Eliminamos latitud y longitud
data_train_with_clusters.to_csv("DIM_tienda_train_with_clusters_cleaned.csv", index
data_test_with_clusters.to_csv("DIM_tienda_test_with_clusters_cleaned.csv", index=F
print("Archivos exportados como 'DIM_tienda_train_with_clusters_cleaned.csv' y 'DIM

colors = ['red', 'blue', 'green', 'purple', 'orange', 'darkred']

## MAPA TRAIN ##
map_center_train = [
    data_train_with_clusters['LATITUD_NUM'].mean(),
    data_train_with_clusters['LONGITUD_NUM'].mean()
]
map_train = folium.Map(location=map_center_train, zoom_start=10)

for _, row in data_train_with_clusters.iterrows():
    color_index = row['cluster'] % len(colors)
    folium.CircleMarker(
        location=[row['LATITUD_NUM'], row['LONGITUD_NUM']],
        radius=5,
        color=colors[color_index],
        fill=True,
```

```
        fill_color=colors[color_index],
        fill_opacity=0.7,
        popup=f"Cluster: {row['cluster']}<br>Lat: {row['LATITUD_NUM']}<br>Lon: {row
    ).add_to(map_train)

map_train.save("clusters_train_map.html")
print("Mapa del conjunto TRAIN guardado como 'clusters_train_map.html'.")

## MAPA TEST ##
map_center_test = [
    data_test_with_clusters['LATITUD_NUM'].mean(),
    data_test_with_clusters['LONGITUD_NUM'].mean()
]
map_test = folium.Map(location=map_center_test, zoom_start=10)

for _, row in data_test_with_clusters.iterrows():
    color_index = row['cluster'] % len(colors)
    folium.CircleMarker(
        location=[row['LATITUD_NUM'], row['LONGITUD_NUM']],
        radius=5,
        color=colors[color_index],
        fill=True,
        fill_color=colors[color_index],
        fill_opacity=0.7,
        popup=f"Cluster: {row['cluster']}<br>Lat: {row['LATITUD_NUM']}<br>Lon: {row
    ).add_to(map_test)

map_test.save("clusters_test_map.html")
print("Mapa del conjunto TEST guardado como 'clusters_test_map.html'.")
```

```
Archivos exportados como 'DIM_tienda_train_with_clusters_cleaned.csv' y 'DIM_tienda_
test_with_clusters_cleaned.csv'.
Mapa del conjunto TRAIN guardado como 'clusters_train_map.html'.
Mapa del conjunto TEST guardado como 'clusters_test_map.html'.
```