# Proyecto 1 - Data Science
# Importación y Análisis de datos


- Paula Barillas - 22764
- Derek Arreaga - 22537
- Mónica Salvatierra - 22249


Link del repositorio: https://github.com/alee2602/PROYECTO1-DS

En este cuaderno documenta la obtención, transformación y exploración de datos obtenidos del sitio web del Mineduc, los cuales describen un grupo de centros de educación en Guatemala.

## 1. Obtención y formato de datos

Los datos obtenidos se encontraban en formato `.xls`, por lo que se realizó el script generate-csvfiles.py, el cúal junto a las librerías **`bs4`** y **`pandas`**, se convierte cada archivo `.xls` a `.csv`.

Desde la carpeta raíz del proyecto

```
cd scripts
python generate-csvfiles.py
```

Luego de eso ya tendremos todos los archivos `.xls` de la carpeta data/raw, convertidos a formato `.csv` en la carpeta data/csv

## 2. Importación y unificación de datos

In [49]:
```
import pandas as pd
import unicodedata
import os
import re

df = pd.read_csv('data/csv/altaverapaz.csv')
df.head()
```

Out[49]:

| | CODIGO | DISTRITO | DEPARTAMENTO | MUNICIPIO | ESTABLECIMIENTO | DIRECCION | TELE |
|---|---|---|---|---|---|---|---|
| **0** | 16-01-0138-46 | 16-031 | ALTA VERAPAZ | COBAN | COLEGIO COBAN | KM.2 SALIDA A SAN JUAN CHAMELCO ZONA 8 | 779 |
| **1** | 16-01-0139-46 | 16-031 | ALTA VERAPAZ | COBAN | COLEGIO PARTICULAR MIXTO VERAPAZ | KM 209.5 ENTRADA A LA CIUDAD | 773 |
| **2** | 16-01-0140-46 | 16-031 | ALTA VERAPAZ | COBAN | COLEGIO "LA INMACULADA" | 7A. AVENIDA 11-109 ZONA 6 | 782 |
| **3** | 16-01-0141-46 | 16-005 | ALTA VERAPAZ | COBAN | ESCUELA NACIONAL DE CIENCIAS COMERCIALES | 2A CALLE 11-10 ZONA 2 | 795 |
| **4** | 16-01-0142-46 | 16-005 | ALTA VERAPAZ | COBAN | INSTITUTO NORMAL MIXTO DEL NORTE 'EMILIO ROSAL... | 3A AVE 6-23 ZONA 11 | 795 |

Primero se importó únicamente **1** archivo para visualizar las columnas que tendrán nuestros archivos, las cuáles son:

**'CODIGO'**, **'DISTRITO'**, **'DEPARTAMENTO'**, **'MUNICIPIO'**, **'ESTABLECIMIENTO'**, **'DIRECCION'**, **'TELEFONO'**, **'SUPERVISOR'**, **'DIRECTOR'**, **'NIVEL'**, **'SECTOR'**, **'AREA'**, **'STATUS'**, **'MODALIDAD'**, **'JORNADA'**, **'PLAN'**, **'DEPARTAMENTAL'**

Ahora importaremos todos los archivos y los concatenaremos en un solo **df** para poder analizarlos correctamente.

In [50]:
```
import pandas as pd
import os

columnas = ['CODIGO', 'DISTRITO', 'DEPARTAMENTO', 'MUNICIPIO', 'ESTABLECIMIENTO',
            'DIRECCION', 'TELEFONO', 'SUPERVISOR', 'DIRECTOR', 'NIVEL', 'SECTOR',
            'AREA', 'STATUS', 'MODALIDAD', 'JORNADA', 'PLAN', 'DEPARTAMENTAL']

carpeta = 'data/csv/'
dfs = []

for archivo in os.listdir(carpeta):
    if archivo.endswith('.csv'):
        ruta_completa = os.path.join(carpeta, archivo)
        temp_df = pd.read_csv(ruta_completa)
        print(f'Leyendo {archivo} con {temp_df.shape[0]} centros educativos')
        dfs.append(pd.read_csv(ruta_completa))

df = pd.concat(dfs, ignore_index=True)
```

```
Leyendo altaverapaz.csv con 294 centros educativos
Leyendo bajaverapaz.csv con 94 centros educativos
Leyendo chimaltenango.csv con 300 centros educativos
Leyendo chiquimula.csv con 136 centros educativos
Leyendo ciudadcapital.csv con 860 centros educativos
Leyendo elprogreso.csv con 97 centros educativos
Leyendo escuintla.csv con 393 centros educativos
Leyendo guatemala.csv con 1036 centros educativos
Leyendo huehuetenango.csv con 295 centros educativos
Leyendo izabal.csv con 273 centros educativos
Leyendo jalapa.csv con 121 centros educativos
Leyendo jutiapa.csv con 296 centros educativos
Leyendo peten.csv con 270 centros educativos
Leyendo quetzaltenango.csv con 365 centros educativos
Leyendo quiche.csv con 184 centros educativos
Leyendo retalhuleu.csv con 272 centros educativos
Leyendo sacatepequez.csv con 206 centros educativos
Leyendo sanmarcos.csv con 431 centros educativos
Leyendo santarosa.csv con 133 centros educativos
Leyendo solola.csv con 111 centros educativos
Leyendo suchitepequez.csv con 296 centros educativos
Leyendo totonicapan.csv con 51 centros educativos
Leyendo zacapa.csv con 70 centros educativos
```

### Obtener valores de columnas generales

In [51]:
```
for col in ['NIVEL', 'SECTOR', 'AREA', 'STATUS', 'MODALIDAD', 'JORNADA', 'PLAN', 'D
    print(f'Datos únicos de la columna {col}:\n{df[col].unique()}')
```

```
Datos únicos de la columna NIVEL:
['DIVERSIFICADO']
Datos únicos de la columna SECTOR:
['PRIVADO' 'OFICIAL' 'MUNICIPAL' 'COOPERATIVA']
Datos únicos de la columna AREA:
['URBANA' 'RURAL' 'SIN ESPECIFICAR']
Datos únicos de la columna STATUS:
['ABIERTA']
Datos únicos de la columna MODALIDAD:
['MONOLINGUE' 'BILINGUE']
Datos únicos de la columna JORNADA:
['MATUTINA' 'VESPERTINA' 'DOBLE' 'NOCTURNA' 'SIN JORNADA' 'INTERMEDIA']
Datos únicos de la columna PLAN:
['DIARIO(REGULAR)' 'FIN DE SEMANA' 'A DISTANCIA' 'SEMIPRESENCIAL'
 'SEMIPRESENCIAL (FIN DE SEMANA)' 'SEMIPRESENCIAL (UN DÍA A LA SEMANA)'
 'VIRTUAL A DISTANCIA' 'SEMIPRESENCIAL (DOS DÍAS A LA SEMANA)' 'SABATINO'
 'INTERCALADO' 'DOMINICAL' 'MIXTO']
Datos únicos de la columna DEPARTAMENTAL:
['ALTA VERAPAZ' 'BAJA VERAPAZ' 'CHIMALTENANGO' 'CHIQUIMULA'
 'GUATEMALA NORTE' 'GUATEMALA ORIENTE' 'GUATEMALA OCCIDENTE'
 'GUATEMALA SUR' 'EL PROGRESO' 'ESCUINTLA' 'HUEHUETENANGO' 'IZABAL'
 'JALAPA' 'JUTIAPA' 'PETÉN' 'QUETZALTENANGO' 'QUICHÉ' 'QUICHÉ NORTE'
 'RETALHULEU' 'SACATEPÉQUEZ' 'SAN MARCOS' 'SANTA ROSA' 'SOLOLÁ'
 'SUCHITEPÉQUEZ' 'TOTONICAPÁN' 'ZACAPA']
```

In [52]:
```
df.shape
```

Out[52]:
```
(6584, 17)
```

# 3. Descripción general del dataset

A continuación, se describen los datos obtenidos del sitio web del Mineduc:

Los datos corresponden a los **6584** centros educativos a nivel nacional que imparten **educación diversificada**. Están organizados en **23 archivos CSV**, uno por cada **departamento** de Guatemala. Cada archivo contiene **17 columnas** con información relevante sobre los establecimientos:

- `CODIGO` : Identificador único del establecimiento, con el formato `XX-XX-XXXX-XX` .

    - **Primer bloque ( `XX` )**: Código del departamento (ej. `18` para Izabal).
    - **Segundo bloque ( `XX` )**: Código del municipio dentro del departamento.
    - **Tercer bloque ( `XXXX` )**: Identificador interno del centro educativo.
    - **Cuarto bloque ( `XX` )**: Es siempre `46` , posiblemente indica el nivel educativo (diversificado) o el sistema de codificación actual del Mineduc.
- `DISTRITO` : Código del distrito escolar al que pertenece el establecimiento. Tiene el formato `XX-XXX` , donde el primer número corresponde al departamento, y los últimos tres identifican las zonas operativas o educativas regionales dentro del departamento. **Ejemplo:** `18-007` y `18-008` son distritos distintos dentro de Izabal.
- `DEPARTAMENTO` : Nombre del departamento (ej. `IZABAL` ).
- `MUNICIPIO` : Municipio donde se ubica el establecimiento (ej. `PUERTO BARRIOS` ).
- `ESTABLECIMIENTO` : Nombre oficial del centro educativo. Puede incluir el tipo ( `COLEGIO` , `ESCUELA` , `INSTITUTO` ) y detalles como mixto, cristiano, etc.
- `DIRECCIÓN` : Ubicación del centro.
- `TELEFONO` : Número telefónico del establecimiento, si está disponible.
- `SUPERVISOR` : Nombre del supervisor distrital asignado al establecimiento.
- `DIRECTOR` : Nombre del director o responsable del centro educativo.
- `NIVEL` : Máximo nivel educativo ofrecido. En este caso todos son `DIVERSIFICADO` .
- `SECTOR` : Define el sector del establecimiento, puede ser:
    - **PRIVADO**
    - **OFICIAL**
    - **MUNICIPAL**
    - **COOPERATIVA**
- `AREA` : Área geográfica según categorización del Mineduc: **URBANA**, **RURAL** o **SIN ESPECIFICAR**.
- `STATUS` : Estado de funcionamiento del centro educativo. En estos datos, el valor para todos es `ABIERTA` .
- `MODALIDAD` : Tipo de modalidad educativa: `MONOLINGUE` o `BILINGUE` .
- `JORNADA` : Jornada en que opera el centro: **MATUTINA**, **VESPERTINA**, **DOBLE**, **NOCTURNA**, **SIN JORNADA**, **INTERMEDIA**.
- `PLAN` : Plan educativo implementado:
    - **DIARIO(REGULAR)**
    - **FIN DE SEMANA**
    - **A DISTANCIA**
    - **SEMIPRESENCIAL**
    - **SEMIPRESENCIAL (FIN DE SEMANA)**
    - **SEMIPRESENCIAL (UN DÍA A LA SEMANA)**
    - **VIRTUAL A DISTANCIA**
    - **SEMIPRESENCIAL (DOS DÍAS A LA SEMANA)**
    - **SABATINO**
    - **INTERCALADO**
    - **DOMINICAL**
    - **MIXTO**
- `DEPARTAMENTAL` : Nombre del departamento al que pertenece el centro educativo. En algunos casos, describe subdivisiones regionales dentro de departamentos con alta densidad de establecimientos, como **GUATEMALA NORTE**, **GUATEMALA ORIENTE**,

**GUATEMALA OCCIDENTE**, **GUATEMALA SUR**, o distinciones como **QUICHÉ** y **QUICHÉ NORTE**.

# 4. Variables que requieren limpieza

## Inspección inicial del estado de los datos

In [53]:
```
# Tamaño y columnas
print("Dimensiones:", df.shape)
print("Columnas:", df.columns.tolist())

# Revisar nulos
print("Valores nulos por columna:\n", df.isnull().sum())

# Tipos de datos
print("Tipos de datos:\n", df.dtypes)

df.sample(5)
```

```
Dimensiones: (6584, 17)
Columnas: ['CODIGO', 'DISTRITO', 'DEPARTAMENTO', 'MUNICIPIO', 'ESTABLECIMIENTO', 'DI
RECCION', 'TELEFONO', 'SUPERVISOR', 'DIRECTOR', 'NIVEL', 'SECTOR', 'AREA', 'STATUS',
'MODALIDAD', 'JORNADA', 'PLAN', 'DEPARTAMENTAL']
Valores nulos por columna:
 CODIGO              0
DISTRITO             0
DEPARTAMENTO         0
MUNICIPIO            0
ESTABLECIMIENTO      0
DIRECCION            2
TELEFONO            45
SUPERVISOR           0
DIRECTOR            23
NIVEL                0
SECTOR               0
AREA                 0
STATUS               0
MODALIDAD            0
JORNADA              0
PLAN                 0
DEPARTAMENTAL        0
dtype: int64
Tipos de datos:
 CODIGO              object
DISTRITO             object
DEPARTAMENTO         object
MUNICIPIO            object
ESTABLECIMIENTO      object
DIRECCION            object
TELEFONO             object
SUPERVISOR           object
DIRECTOR             object
NIVEL                object
SECTOR               object
AREA                 object
STATUS               object
MODALIDAD            object
JORNADA              object
PLAN                 object
DEPARTAMENTAL        object
dtype: object
```

Out[53]:

| | CODIGO | DISTRITO | DEPARTAMENTO | MUNICIPIO | ESTABLECIMIENTO | |
|---|---|---|---|---|---|---|
| **5190** | 11-03-0022-46 | 11-018 | RETALHULEU | SANTA CRUZ MULUA | COLEGIO MIXTO SANTA CRUZ | 2 5TA |
| **3373** | 13-09-0046-46 | 13-022 | HUEHUETENANGO | SAN ILDEFONSO IXTAHUACAN | COLEGIO EVANGÉLICO MIXTO PRIVADO "EMANUEL" | |
| **2995** | 01-15-0759-46 | 01-411 | GUATEMALA | VILLA NUEVA | LICEO TECNOLÓGICO MAHANAIM | |
| **3779** | 21-01-0104-46 | 21-002 | JALAPA | JALAPA | INSTITUTO NORMAL CENTROAMERICANO PARA VARONES | CA ROJA 2, |
| **4509** | 09-01-0334-46 | 09-006 | QUETZALTENANGO | QUETZALTENANGO | COLEGIO PRIVADO "BELLAS ARTES" | QUE |

Como se puede observar, todas las columnas son de tipo `object`, por lo que para la limpieza de todas las columnas se eliminarán espacios extremos y convertir a mayúsculas.

## Limpiar columnas tipo texto: eliminar espacios extremos y convertir a mayúsculas

In [54]:
```
for col in df.select_dtypes(include="object").columns:
    df[col] = df[col].astype(str).str.strip().str.upper()
```

En este paso se estandarizó todo el texto de las columnas tipo cadena. Esto se realizó con el propósito de evitar inconsistencias causadas por diferencias en mayúsculas y minúsculas, y por espacios innecesarios al inicio o final de los campos. Esta limpieza permite mejorar la detección de duplicados, corregir errores ortográficos y facilitar futuras agrupaciones.

Ahora con todas las columnas sin inconcistencias por diferencia de caracteres, se procede a analizar todas las columnas una por una.

# 1. CÓDIGO

**Ver ejemplos distintos de códigos (únicos)**

In [55]:
```
df["CODIGO"].sample(20, random_state=1)
```

Out[55]:
```
5161    11-01-1406-46
2805    01-14-0187-46
3327    13-04-0112-46
3531    18-01-0320-46
6333    10-08-0026-46
3821    21-01-0417-46
4152    22-14-0080-46
1851    05-01-0459-46
6414    10-15-0040-46
4523    09-01-0384-46
5442    03-09-0056-46
1793    05-01-0253-46
1584    00-18-0223-46
1691    02-01-0062-46
5891    12-29-0029-46
3172    01-17-0315-46
1988    05-05-0077-46
4179    22-16-0031-46
2851    01-15-0119-46
552     04-03-0260-46
Name: CODIGO, dtype: object
```

**Ver longitud del identificador de cada establecimiento**

In [56]:
```
df["LARGO_CODIGO"] = df["CODIGO"].astype(str).str.len()
df["LARGO_CODIGO"].value_counts().sort_index()
```

Out[56]:
```
LARGO_CODIGO
13    6584
Name: count, dtype: int64
```

Se evaluó la estructura de los códigos institucionales para garantizar su consistencia. Esto permitió detectar posibles errores de codificación o registros atípicos. No existen variaciones en el formato ni en la cantidad de dígitos en cada código.

In [57]:
```
df = df.drop("LARGO_CODIGO", axis=1)
```

## 2. DISTRITO

**Ver ejemplos distintos de códigos de cada distrito (únicos)**

In [58]:
```
df["DISTRITO"].sample(20, random_state=1)
```

Out[58]:
```
5161    11-017
2805    01-411
3327    13-012
3531    18-008
6333    10-009
3821    21-004
4152    22-023
1851    05-033
6414    10-021
4523    09-006
5442    03-006
1793    05-033
1584    01-403
1691    02-021
5891    12-053
3172    01-641
1988    05-031
4179    22-026
2851    01-214
552     04-033
Name: DISTRITO, dtype: object
```

**Ver longitud del identificador de cada distrito**

In [59]:
```
df["LARGO_DISTRITO"] = df["DISTRITO"].astype(str).str.len()
df["LARGO_DISTRITO"].value_counts().sort_index()
```

Out[59]:
```
LARGO_DISTRITO
6    6584
Name: count, dtype: int64
```

Se evaluó la estructura de los identificadores de cada distrito para garantizar su consistencia. Esto permitió detectar posibles errores de codificación o registros atípicos. No existen variaciones en el formato ni en la cantidad de dígitos en cada identificador del distrito

In [60]:
```
df = df.drop("LARGO_DISTRITO", axis=1)
```

## 3. DEPARTAMENTO

**Eliminar espacios inconsistentes y cualquier acentuación**

In [61]:
```
def limpiar_departamento(nombre):
    nombre = unicodedata.normalize("NFKD", nombre)
    nombre = ''.join(c for c in nombre if not unicodedata.combining(c))
    return nombre

df["DEPARTAMENTO"] = df["DEPARTAMENTO"].apply(limpiar_departamento)
```

**Obtener los valores únicos**

In [62]:
```
print("Departamentos únicos encontrados:")
print(df["DEPARTAMENTO"].sort_values().unique())
```

```
Departamentos únicos encontrados:
['ALTA VERAPAZ' 'BAJA VERAPAZ' 'CHIMALTENANGO' 'CHIQUIMULA'
 'CIUDAD CAPITAL' 'EL PROGRESO' 'ESCUINTLA' 'GUATEMALA' 'HUEHUETENANGO'
 'IZABAL' 'JALAPA' 'JUTIAPA' 'PETEN' 'QUETZALTENANGO' 'QUICHE'
 'RETALHULEU' 'SACATEPEQUEZ' 'SAN MARCOS' 'SANTA ROSA' 'SOLOLA'
 'SUCHITEPEQUEZ' 'TOTONICAPAN' 'ZACAPA']
```

**Validar contra la lista de departamentos**

Tanto para los departamentos como para los municipios, se extrajeron datos del Instituto Nacional de Estadística Guatemala para comparar los valores de las columnas con los valores reales.

En el caso de los departamentos sabemos que `CIUDAD CAPITAL` es tomada como un departamento debido a la gran cantidad de insituciones que tiene, por lo que lo tomaremos como válido.

In [63]:
```
departamentos_oficiales = pd.read_csv('data/extra/departamentos.csv')["DEPARTAMENTO

departamentos_oficiales.append("CIUDAD CAPITAL")

departamentos_encontrados = set(df["DEPARTAMENTO"].unique())
departamentos_invalidos = departamentos_encontrados - set(departamentos_oficiales)

print(" Departamentos no oficiales encontrados:")
print(departamentos_invalidos)
```

```
 Departamentos no oficiales encontrados:
set()
```

Se normalizó el campo `DEPARTAMENTO` mediante la conversión a mayúsculas, eliminación de espacios y tildes. Posteriormente, se comparó contra la lista oficial de departamentos de Guatemala para detectar valores inválidos. Los errores se corrigen si se deben a variantes tipográficas; si no, se analizan individualmente.

# MUNICIPIO

## Eliminar espacios inconsistentes y cualquier acentuación

In [64]:
```
def limpiar_municipio(nombre):
    nombre = unicodedata.normalize("NFKD", nombre)
    nombre = ''.join(c for c in nombre if not unicodedata.combining(c))
    nombre = " ".join(nombre.split())
    return nombre

df["MUNICIPIO"] = df["MUNICIPIO"].apply(limpiar_municipio)
```

## Revisar valores únicos

In [65]:
```
print("Cantidad de municipios distintos:", df["MUNICIPIO"].nunique())
print(df["MUNICIPIO"].sort_values().unique()[:30])
```

```
Cantidad de municipios distintos: 343
['ACATENANGO' 'AGUA BLANCA' 'AGUACATAN' 'ALOTENANGO' 'AMATITLAN'
 'ANTIGUA GUATEMALA' 'ASUNCION MITA' 'ATESCATEMPA' 'AYUTLA' 'BARBERENA'
 'CABANAS' 'CABRICAN' 'CAJOLA' 'CAMOTAN' 'CANILLA' 'CANTEL' 'CASILLAS'
 'CATARINA' 'CHAHAL' 'CHAJUL' 'CHAMPERICO' 'CHIANTLA' 'CHICACAO'
 'CHICAMAN' 'CHICHE' 'CHIMALTENANGO' 'CHINAUTLA' 'CHINIQUE' 'CHIQUIMULA'
 'CHIQUIMULILLA']
```

**Validar contra la lista de municipios**

Se utilizarán nuevamente los datos del Instituto Nacional de Estadística Guatemala para comparar los valores de las columnas con los valores reales.

In [66]:
```
municipios_oficiales = pd.read_csv('data/extra/municipios.csv')["MUNICIPIO"].str.up


municipios_encontrados = set(df['MUNICIPIO'].unique()) - {mun for mun in df['MUNICI
municipios_invalidos = municipios_encontrados - set(municipios_oficiales)

print(" Municipios no oficiales encontrados:")
print(municipios_invalidos)
```

```
 Municipios no oficiales encontrados:
{'PETATAN', 'PACHALUN', 'LA TINTA'}
```

Al comparar los municipios del resultado con los del listado oficial, se encontró que es debido a faltas otrográficas o por ser la versión corta del nombre del municipio:

| Valor encontrado | Valor Real |
|---|---|
| PETATAN | SANTIAGO PETATAN |
| LA TINTA | SANTA CATALINA LA TINTA |
| PACHALUN | PACHALUM |

Para la limpieza se recomienda reemplazar por el nombre completo para mantener información oficial valiosa.

**Ver municipios repetidos escritos diferente dentro de un mismo departamento**

In [67]:
```
# Agrupar por DEPARTAMENTO y MUNICIPIO
municipios_por_depto = df.groupby(["DEPARTAMENTO", "MUNICIPIO"]).size().reset_index

# Ver si hay municipios con nombres similares dentro del mismo departamento
print("Total de combinaciones DEPARTAMENTO-MUNICIPIO:", len(municipios_por_depto))
```

```
Total de combinaciones DEPARTAMENTO-MUNICIPIO: 348
```

**Verificar combinaciones de municipios con departamentos**

In [68]:
```
import pandas as pd

# Cargar DataFrames
municipios_oficiales = pd.read_csv('data/extra/municipios.csv')
departamentos_oficiales = pd.read_csv('data/extra/departamentos.csv')
```

```
# Normalizar DEPARTAMENTO en instituciones (quitar acentos, mayúsculas)
import unicodedata
df['DEPARTAMENTO'] = df['DEPARTAMENTO'].apply(
    lambda x: ''.join(c for c in unicodedata.normalize('NFKD', str(x)) if not unico
).str.upper()

# Extraer códigos de departamento y municipio del CODIGO (16-01-0138-46 -> 1601)
df['CODIGO_DEP_MUN'] = df['CODIGO'].str.split('-').str[0] + df['CODIGO'].str.split(

# Crear diccionario de códigos a departamentos y municipios
dep_map = dict(zip(departamentos_oficiales['CODIGO'].astype(str).str.zfill(2), depa
mun_map = dict(zip(municipios_oficiales['CODIGO'].astype(str).str.zfill(4), municip

# Verificar combinaciones
def verificar_combinacion(row):
    cod_dep = row['CODIGO_DEP_MUN'][:2]  # Primeros 2 dígitos (departamento)
    cod_mun = row['CODIGO_DEP_MUN']  # Los 4 dígitos (departamento + municipio)
    dep = row['DEPARTAMENTO']
    mun = row['MUNICIPIO'].upper()

    # Caso especial: CIUDAD CAPITAL
    if dep == 'CIUDAD CAPITAL' and mun.startswith('ZONA'):
        return True

    # Verificar departamento
    dep_oficial = dep_map.get(cod_dep)
    if dep_oficial != dep:
        return False

    # Verificar municipio
    mun_oficial = mun_map.get(cod_mun)
    return mun_oficial == mun

# Identificar combinaciones incorrectas
df['ES_VALIDO'] = df.apply(verificar_combinacion, axis=1)
df_errores = df[~df['ES_VALIDO']].copy()

df_errores.to_csv('limpieza/errores_combinaciones.csv', index=False, encoding='utf-
print(f"Instituciones con combinaciones incorrectas guardadas en 'data/errores_comb
```

```
Instituciones con combinaciones incorrectas guardadas en 'data/errores_combinacione
s.csv': 31
```

Se normalizaron los municipios, eliminando tildes y espacios innecesarios. Esto permitió identificar errores ortográficos comunes y garantizar consistencia entre registros. Luego se validó que cada municipio pertenezca al listado oficial de municipios oficiales.

En el caso de los municipios llamados `ZONA N`, sabemos que son de `CIUDAD CAPITAL` por lo que se realizó una condición especial para esos valores.

De último se verificaron las combinaciones de `MUNICIPIO` + `DEPARTAMENTO` para asegurarse de que los valores asignados tengan sentido. Como resultado se obtuvieron **31** filas con error de combinación, al analizar el csv generado (errores_combinaciones.csv) se encontró que los únicos errores era para los mismos municipios que tienen faltas ortográficas.

## ESTABLECIMIENTO

**Crear una columna auxiliar ESTABLECIMIENTO_LIMPIO**

Su objetivo es estandarizar nombres para facilitar detección de duplicados y variantes de una misma institución, colegio, entre otros.

In [69]:
```
def limpiar_nombre_establecimiento(nombre):
    nombre = str(nombre).strip().upper()

    # Eliminar todas las comillas dobles y simples en cualquier parte
    nombre = nombre.replace('"', '').replace("'", "")
```

```
    # Quitar espacios múltiples
    nombre = " ".join(nombre.split())

    return nombre

df["ESTABLECIMIENTO_LIMPIO"] = df["ESTABLECIMIENTO"].apply(limpiar_nombre_estableci

print(df[["ESTABLECIMIENTO", "ESTABLECIMIENTO_LIMPIO"]].drop_duplicates().sample(10
```

```
                                    ESTABLECIMIENTO  \
4495                              COLEGIO JOHN HARVARD
2239                  LICEO INMACULADO CORAZÓN DE MARÍA
6288  INSTITUTO PRIVADO MIXTO PROFESIONAL  VISION ED...
4550  CENTRO DE EDUCACIÓN EXTRAESCOLAR -CEEX- QUETZA...
2578                                LICEO BUENOS AIRES
3627              INSTITUTO PRIVADO MIXTO "SAN ANTONIO"
4161                 COLEGIO PARTICULAR MIXTO  SAN JOSE
254           INSTITUTO TECNICO VOCACIONAL  SAN AGUSTIN
3579           PROGRAMA DE EDUCACION ALTERNATIVA PRONEA
972      INSTITUTO TECNOLÓGICO DIGITALES COMERCIALES II

                                ESTABLECIMIENTO_LIMPIO
4495                              COLEGIO JOHN HARVARD
2239                  LICEO INMACULADO CORAZÓN DE MARÍA
6288  INSTITUTO PRIVADO MIXTO PROFESIONAL VISION EDU...
4550  CENTRO DE EDUCACIÓN EXTRAESCOLAR -CEEX- QUETZA...
2578                                LICEO BUENOS AIRES
3627               INSTITUTO PRIVADO MIXTO SAN ANTONIO
4161                 COLEGIO PARTICULAR MIXTO SAN JOSE
254            INSTITUTO TECNICO VOCACIONAL SAN AGUSTIN
3579           PROGRAMA DE EDUCACION ALTERNATIVA PRONEA
972      INSTITUTO TECNOLÓGICO DIGITALES COMERCIALES II
```

## Visualizar algunos ejemplos de variaciones en nombres de establecimientos para su estandarización

In [70]:
```
#%pip install rapid fuzz
```

In [71]:
```
from rapidfuzz import process, fuzz

# Obtener todos los nombres únicos
nombres_unicos = df["ESTABLECIMIENTO_LIMPIO"].dropna().unique()

# Diccionario para guardar grupos similares
grupos_similares = {}

# Umbral de similitud
umbral = 90

# Recorrer nombres y buscar similares
for nombre in nombres_unicos:
    similares = process.extract(nombre, nombres_unicos, scorer=fuzz.token_sort_rati
    # Filtrar solo los que superan el umbral y no sean el mismo nombre exacto
    similares = [s for s, score, _ in similares if s != nombre and score >= umbral]
    if similares:
        grupos_similares[nombre] = similares

# Mostrar ejemplos de grupos similares encontrados
for i, (base, grupo) in enumerate(grupos_similares.items()):
    print(f"\nGrupo {i+1}:")
    print(f"→ Base: {base}")
    for g in grupo:
        print(f"   - {g}")
    if i == 4:  # Mostrar solo primeros 5 grupos
        break
```

```
Grupo 1:
→ Base: ESCUELA NACIONAL DE CIENCIAS COMERCIALES
   - ESCUELA NACIONAL DE CIENCIAS COMERCIALES NO.2
   - ESCUELA NACIONAL DE CIENCIAS COMERCIALES NO.5
   - ESCUELA NACIONAL DE CIENCIAS COMERCIALES NO. 3
   - ESCUELA NACIONAL CENTRAL DE CIENCIAS COMERCIALES
   - ESCUELA NACIONAL DE CIENCIAS COMERCIALES AMERICA
   - ESCUELA DE CIENCIAS COMERCIALES NOCTURNA

Grupo 2:
→ Base: INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADA
   - INSTITUTO NACIONAL DE EDUCACIÓN DIVERSIFICADA
   - INSTITUTO NACIONAL DE EDUCACIÒN DIVERSIFICADA
   - INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADO
   - INSTITUTO NACIONAL DE EDUCACIÓN DIVERSIFICADO
   - INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADA INED
   - INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADA, INED
   - INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADA ITZAPA
   - INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADA -INED-
   - INSTITUTO NACIONAL DE EDUCACION DIVERSIFICADA PUENTE

Grupo 3:
→ Base: COLEGIO CRISTIANO BILINGUE VERBO
   - COLEGIO CRISTIANO BILINGÜE VERBO

Grupo 4:
→ Base: INSTITUTO MIXTO DE EDUCACION BILINGUE INTERCULTURAL -IMEBI-
   - INSTITUTO MIXTO DE EDUCACIÓN BILINGÜE INTERCULTURAL -IMEBI-

Grupo 5:
→ Base: COLEGIO PRIVADO MIXTO TECNOLÓGICO EN INFORMÁTICA
   - COLEGIO PRIVADO MIXTO TECNOLÓGICO EN INFORMATICA
   - COLEGIO PRIVADO MIXTO TECNOLOGICO EN INFORMATICA
```

Como se puede observar, si hay algunos nombres de establecimientos que están escritos de diferente manera. Es decir, hay variaciones en cuanto a tíldes, espacios, uso de guiones, entre otros caracteres para representar a un mismo establecimiento, independientemente de si se encuentra en algún otro departamento o municipio. Por ende, se busca generar un csv para poder observar de cerca todas esas variaciones y estandarizar el nombre de los establecimientos.

In [72]:
```python
# Convertir los grupos a lista de pares
pares = []
for base, similares in grupos_similares.items():
    for s in similares:
        pares.append((base, s))

df_similares = pd.DataFrame(pares, columns=["Nombre_Base", "Nombre_Similar"])
df_similares.to_csv("limpieza/establecimientos_similares.csv", index=False)
print("Archivo exportado como 'establecimientos_similares.csv'")
```

```
Archivo exportado como 'establecimientos_similares.csv'
```

Al analizar varios de los grupos, se puede notar que hay varias faltas ortográficas que se suelen repetir como las siguientes:

| Falta | Correcto |
|---|---|
| "EDCACION" | "EDUCACION" |
| "NACIONA" | "NACIONAL" |
| "COLEGO" | "COLEGIO" |

Por ello, en la limpieza se deberá de hacer un `.replace()` para corregir las faltas usuales.

# DIRECCIÓN

In [73]:
```python
df["DIRECCION"].head(10)
```

Out[73]:
```
0              KM.2 SALIDA A SAN JUAN CHAMELCO ZONA 8
1                        KM 209.5 ENTRADA A LA CIUDAD
2                           7A. AVENIDA 11-109 ZONA 6
3                               2A CALLE 11-10 ZONA 2
4                                 3A AVE 6-23 ZONA 11
5                                5A. CALLE 1-9 ZONA 3
6                               11 AVENIDA 5-17 ZONA 4
7    DIAGONAL 08 8-05 ZONA 8, BARRIO CANTÓN LAS CASAS
8                                   12 AV. 2-12 ZONA 1
9                               5TA. CALLE 2-23 ZONA 4
Name: DIRECCION, dtype: object
```

Como se puede observar a simple vista, hay distintas maneras de escribir direcciones, en primer lugar para escribir **KM**, algunos lo escriben con un `.` después de la abreviatura. El mismo caso con las **Avenidas**. Es por eso que se planea realizar un proceso de limpieza y estandarización conservador para asegurarse no cambiar mucho las direcciones.

# TELEFONO

In [74]:
```
df["TELEFONO"].dropna().str.len().unique()
```

Out[74]:
```
array([ 8,  3, 17,  7,  6, 26, 10,  9,  2,  1, 18, 16])
```

In [75]:
```
for length in df["TELEFONO"].dropna().str.len().unique():
    print(f"Longitud {length}:", df[df["TELEFONO"].str.len() == length]["TELEFONO"]
```

```
Longitud 8: ['77945104', '77367402', '78232301', '79514215', '79521468', '57101061',
'79522555', '77930045', '79545566', '79514754', '79514754', '79521468', '79521468',
'79522793', '51922050', '79510413', '79511103', '77945181', '53530198', '79417300',
'79510413', '30367501', '57565646', '77259205', '57655184', '53331646', '77367652',
'30621374', '79514754', '57101061', '79521190', '33509500', '48979726', '79529830',
'77930803', '55564568', '51922050', '45587767', '79513898', '79521302', '77256954',
'79521765', '79511459', '59584790', '79529830', '79513896', '79417346', '37058145',
'33857209', '33622052']
Longitud 3: ['NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN',
'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN',
'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN',
'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN', 'NAN']
Longitud 17: ['79504027-79504028', '79540830-79540909', '79540830-79540909', '796496
96-78739432', '78739432-79649696', '78739432-79649696', '78739432-79649696', '788804
29-78880419', '24342036-24123888', '24314989-24310955', '24314989-24310955', '663558
87-66312603', '79484391-41051885', '45261770-46007362', '79480085-50307050', '794800
85-50307050', '79524680-79524680', '53886336-53886336', '79497500-78209540', '310894
18-53230377', '42601313-79476691', '42601313-79476691', '53660792-55969273', '536607
92-55969273', '79477890-79477891', '79224958-79223033', '31193946-46843572', '526600
00-78323083', '56769964-77663283', '77663283-77663283', '77375333-42470723', '773753
33-42470723']
Longitud 7: ['4085613', '4225675', '2232068', '2223228', '2232379', '5899624', '2437
011', '5510735', '5364736', '5388633', '4167076', '3127247']
Longitud 6: ['783928', '3033.0', '533290']
Longitud 26: ['78392709-78396701-78396702']
Longitud 10: ['79422150.0', '79420395.0', '41942927.0', '79420290.0', '79420116.0',
'79425959.0', '59799216.0', '79420425.0', '79420328.0', '79420232.0', '79422349.0',
'41007884.0', '79424111.0', '79421747.0', '44043628.0', '41795225.0', '79425959.0',
'37590030.0', '44043628.0', '79422500.0', '79420290.0', '79425429.0', '41795225.0',
'58123452.0', '42415701.0', '55543693.0', '31712671.0', '40104955.0', '56170043.0',
'41007884.0', '79424111.0', '41795225.0', '79421528.0', '53947190.0', '58123452.0',
'55974817.0', '79427427.0', '79420425.0', '44043628.0', '79427427.0', '79424111.0',
'51309580.0', '42580713.0', '51989697.0', '33874323.0', '55223225.0', '79465751.0',
'79465395.0', '79465220.0', '79465220.0']
Longitud 9: ['4215928.0', '7844007.0', '7844007.0', '4140069.0', '5899378.0']
Longitud 2: ['40']
Longitud 1: ['0']
Longitud 18: ['4210394058993785.0', '4210394058993780.0', '4281099156375310.0']
Longitud 16: ['56769964-7766328']
```

**Estrategia de limpieza de teléfonos**

- **Conversión a cadena y estandarización de nulos:**
  Convertir los valores a cadenas. Los valores vacíos o con contenido no numérico (como

`'NAN'`) tratarlos como `"NO DISPONIBLE"`.

- **Separación por delimitadores comunes:**
  Para identificar múltiples teléfonos en una misma celda, utilizar delimitadores comunes como guiones (`-`), barras (`/`) o comas (`,`), para separar así posibles números concatenados.

- **Eliminación de caracteres no numéricos:**
  Limpiar todos los caracteres no numéricos de cada fragmento, para conservar únicamente los dígitos. Esto incluye la eliminación de guiones, espacios, puntos decimales (`.0`) y otros símbolos.

- **Normalización y validación por longitud:**
  Considerar como válidos solo los números con exactamente 8 dígitos (formato estándar en Guatemala).

  - Si un número tiene más de 8 dígitos pero menos de 13, se deben de conservar los últimos 8.
  - Si un fragmento tiene menos de 8 o más de 12 dígitos, descartarlo como inválido.

- **Asignación a columnas:**
  Extraer hasta tres números válidos por registro y se asignaron a las columnas `TELEFONO_LIMPIO`, `TELEFONO_2_LIMPIO` y `TELEFONO_3_LIMPIO`.
  Si no se encuentran números válidos, se debe de registrar como `"NO DISPONIBLE"` en todas las columnas correspondientes.

## SUPERVISOR

**Limpieza de espacios y acentuaciones en el nombre de los supervisores, además de una validación de posibles valores nulos**

In [76]:
```
def limpiar_supervisor(nombre):
    if pd.isnull(nombre):
        return "NO REGISTRADO"


    # Eliminar tildes y caracteres especiales
    nombre = unicodedata.normalize('NFKD', nombre)
    nombre = ''.join(c for c in nombre if not unicodedata.combining(c))

    nombre = re.sub(r'\s+', ' ', nombre)

    return nombre

df["SUPERVISOR_LIMPIO"] = df["SUPERVISOR"].apply(limpiar_supervisor)
```

**Obtener nombres únicos de cada supervisor**

In [77]:
```
print("Supervisores distintos:", df["SUPERVISOR_LIMPIO"].nunique())
df["SUPERVISOR_LIMPIO"].value_counts().head(10)
```

```
Supervisores distintos: 590
```

Out[77]:
```
SUPERVISOR_LIMPIO
MIGUEL ANGEL ARMAS ROCHA                          190
CARLOS HUMBERTO GONZALEZ DE LEON                  171
JUAN ENRIQUE MARTINEZ SOLANO                      106
REMY ARTURO SINAY GUDIEL                           94
ELSA YADIRA MARTINEZ PEREZ                         84
IDALIA DEL ROSARIO LOPEZ SANDOVAL DE PAIZ          81
MILTON ALONSO ALVAREZ FUENTES                      81
ELENA ELIZABETH SUCHITE GARNICA DE QUINTANILLA     78
LUDVIN RICARDO URRUTIA LORENTI                     77
JUAN FRANCISCO GODOY DAVILA                        74
Name: count, dtype: int64
```

**Aplicar Fuzzy matching para detectar nombres parecidos y descartar errores tipográficos**

In [78]:
```
nombres_unicos = df["SUPERVISOR_LIMPIO"].dropna().unique()

grupos_similares = []

for nombre in nombres_unicos:
    similares = process.extract(nombre, nombres_unicos, scorer=fuzz.token_sort_rati
    for s, score, _ in similares:
        if s != nombre and score >= 90:
            grupos_similares.append((nombre, s, score))

df_fuzzy_supervisores = pd.DataFrame(grupos_similares, columns=["SUPERVISOR_BASE",
df_fuzzy_supervisores = df_fuzzy_supervisores.drop_duplicates(subset=["SUPERVISOR_B
```

In [79]:
```
df_fuzzy_supervisores.to_csv("limpieza/supervisores_similares.csv", index=False)
print("Archivo exportado como 'supervisores_similares.csv'")
```

```
Archivo exportado como 'supervisores_similares.csv'
```

Para los supervisores se notó que hay nombres repetidos muy similares, los cuales pueden ser un error de ingreso de datos. Es por eso que se recomienda mediante un diccionario, reemplazar los valores con faltas por los nombres bien escritos.

## DIRECTOR

**Limpieza de espacios y acentuaciones en el nombre de los directores, además de una validación de posibles valores nulos**

In [80]:
```
def limpiar_director(nombre):
    if pd.isnull(nombre):
        return "NO REGISTRADO"


    # Eliminar tildes y caracteres especiales
    nombre = unicodedata.normalize('NFKD', nombre)
    nombre = ''.join(c for c in nombre if not unicodedata.combining(c))

    nombre = re.sub(r'\s+', ' ', nombre)

    return nombre

df["DIRECTOR_LIMPIO"] = df["DIRECTOR"].apply(limpiar_director)
```

**Obtener nombres únicos de cada supervisor**

In [81]:
```
print("Directores distintos:", df["DIRECTOR_LIMPIO"].nunique())
df["DIRECTOR_LIMPIO"].value_counts().head(10)
```

```
Directores distintos: 3816
```

Out[81]:
```
DIRECTOR_LIMPIO
NAN                                23
MARIA DOLORES PEREZ TUCHAN         10
HECTOR REYNALDO GOMEZ AGUILAR       9
SONIA JOSEFINA MORALES CAXAJ        8
MARCO TULIO VARGAS ALVARADO         8
FRANCISCO REVOLORIO LOPEZ           8
RONI ERNESTO RECINOS ESTRADA        8
SILVIA MARLENI ALVARADO GUARDADO    8
MELVIN RAFAEL REYES LOPEZ           8
SANDRA NOEMI ORTIZ ESTRADA          7
Name: count, dtype: int64
```

**Aplicar Fuzzy matching para detectar nombres parecidos y descartar errores tipográficos**

In [82]:
```
df["DIRECTOR_LIMPIO"] = df["DIRECTOR_LIMPIO"].str.strip()
df["DIRECTOR_LIMPIO"] = df["DIRECTOR_LIMPIO"].str.replace("`", "'", regex=False)
df["DIRECTOR_LIMPIO"] = df["DIRECTOR_LIMPIO"].str.replace('"', '', regex=False)
df["DIRECTOR_LIMPIO"] = df["DIRECTOR_LIMPIO"].str.replace(" +", " ", regex=True)
```

In [83]:
```
nombres_unicos = df["DIRECTOR_LIMPIO"].dropna().unique()

grupos_similares = []

for nombre in nombres_unicos:
    similares = process.extract(nombre, nombres_unicos, scorer=fuzz.token_sort_rati
    for s, score, _ in similares:
        if s != nombre and score >= 90:
            grupos_similares.append((nombre, s, score))

df_fuzzy_supervisores = pd.DataFrame(grupos_similares, columns=["DIRECTOR_BASE", "D
df_fuzzy_supervisores = df_fuzzy_supervisores.drop_duplicates(subset=["DIRECTOR_BAS
```

In [84]:
```
df_fuzzy_supervisores.to_csv("limpieza/directores_similares.csv", index=False)
print("Archivo exportado como 'directores_similares.csv'")
```

```
Archivo exportado como 'directores_similares.csv'
```

In [85]:
```
df_fuzzy_supervisores.head()
```

Out[85]:

| | DIRECTOR_BASE | DIRECTOR_SIMILAR | SIMILITUD |
|---|---|---|---|
| **0** | MARTA NORIEGA | MARTA NOREIGA | 92.307692 |
| **1** | MARTA NOREIGA | MARTA NORIEGA | 92.307692 |
| **2** | BRAYAN ONELL GAITAN RODRIGUEZ | BRYAN O'NELL GAITAN RODRIGUEZ | 96.551724 |
| **3** | BRYAN O'NELL GAITAN RODRIGUEZ | BRAYAN ONELL GAITAN RODRIGUEZ | 96.551724 |
| **4** | JULIO ENRIQUE JUAREZ SILVESTRE | JULIO ENRIQUE JAREZ SILVESTRE | 98.305085 |

Al igual que con los directores, también se encontraron varios nombres de supervisores repetidos muy similares, los cuales pueden ser un error de ingreso de datos. Es por eso que se recomienda mediante un diccionario, reemplazar los valores con faltas por los nombres bien escritos..

## Variables Categóricas

Se recorren las columnas categóricas para obtener los valores distintos de cada una.

In [86]:
```
variables_categoricas = {"NIVEL","SECTOR","AREA","STATUS","MODALIDAD","JORNADA","PL

for col in variables_categoricas:
    # Verificar valores distintos
    print(f"Valores distintos para: {col} : {df[col].nunique()}")
    print(df[col].value_counts().head(27))
```

```
Valores distintos para: PLAN : 12
PLAN
DIARIO(REGULAR)                          4039
FIN DE SEMANA                            1308
SEMIPRESENCIAL (FIN DE SEMANA)            489
SEMIPRESENCIAL (UN DÍA A LA SEMANA)       409
A DISTANCIA                               116
SEMIPRESENCIAL                             99
SEMIPRESENCIAL (DOS DÍAS A LA SEMANA)      62
VIRTUAL A DISTANCIA                        52
SABATINO                                    5
INTERCALADO                                 2
DOMINICAL                                   2
MIXTO                                       1
Name: count, dtype: int64
Valores distintos para: AREA : 3
AREA
URBANA             5237
RURAL              1346
SIN ESPECIFICAR       1
Name: count, dtype: int64
Valores distintos para: STATUS : 1
STATUS
ABIERTA    6584
Name: count, dtype: int64
Valores distintos para: JORNADA : 6
JORNADA
DOBLE          1952
VESPERTINA     1834
MATUTINA       1681
SIN JORNADA    1000
NOCTURNA        104
INTERMEDIA       13
Name: count, dtype: int64
Valores distintos para: MODALIDAD : 2
MODALIDAD
MONOLINGUE    6375
BILINGUE       209
Name: count, dtype: int64
Valores distintos para: SECTOR : 4
SECTOR
PRIVADO        5403
OFICIAL         874
COOPERATIVA     213
MUNICIPAL        94
Name: count, dtype: int64
Valores distintos para: DEPARTAMENTAL : 26
DEPARTAMENTAL
GUATEMALA OCCIDENTE    551
GUATEMALA NORTE        536
GUATEMALA SUR          528
SAN MARCOS             431
ESCUINTLA              393
QUETZALTENANGO         365
CHIMALTENANGO          300
SUCHITEPÉQUEZ          296
JUTIAPA                296
HUEHUETENANGO          295
ALTA VERAPAZ           294
GUATEMALA ORIENTE      281
IZABAL                 273
RETALHULEU             272
PETÉN                  270
SACATEPÉQUEZ           206
QUICHÉ                 144
CHIQUIMULA             136
SANTA ROSA             133
JALAPA                 121
SOLOLÁ                 111
EL PROGRESO             97
BAJA VERAPAZ            94
ZACAPA                  70
TOTONICAPÁN             51
```

```
QUICHÉ NORTE            40
Name: count, dtype: int64
Valores distintos para: NIVEL : 1
NIVEL
DIVERSIFICADO    6584
Name: count, dtype: int64
```

Al analizar las distintas categorías de cada columna, se encontró que no hay categorías mal escritas o incongruentes. Como única observación, se recomienda estandarizar todos los valores para evitar tildes o caracteres especiales.

# Plan de Acción

Tras analizar detenidamente todos los datos, se llegó a una conclusión de los pasos que consideramos necesarios para generar un dataset lo más limpio posible.

Así se planificará y ejecutará el proceso completo de limpieza de datos, con el objetivo de estandarizar todas las columnas del dataset original para su posterior análisis:

1. **Unificación de datos:** Se unificarán los datasets utilizando el script `union_datos.py`, consolidando toda la información en un único archivo base.
2. **Duplicación de columnas:** Se duplicarán las columnas originales para aplicar procesos de limpieza sin perder los valores crudos.
3. **Limpieza de columnas de texto:** Se eliminarán espacios en blanco al inicio y al final de las cadenas, y se convertirán los textos a mayúsculas. Para los campos `DEPARTAMENTO` y `MUNICIPIO`, se verificará que sus valores correspondan con los datos oficiales del Instituto Nacional de Estadística de Guatemala.
4. **Normalización de nombres de establecimientos:** En la columna `ESTABLECIMIENTO` se eliminarán caracteres especiales como comillas ( `"` , `'` ) y guiones ( `-` ), y se corregirán errores ortográficos comunes.
5. **Estandarización de direcciones:** Dado que las direcciones suelen presentar un alto grado de variabilidad, se aplicará una estrategia conservadora de limpieza para unificar la nomenclatura de calles, avenidas y otros elementos comunes.
6. **Procesamiento de teléfonos:** Se detectarán y tratarán valores con más de 8 dígitos. En casos particulares (por ejemplo, cadenas de 17 o 26 dígitos), se extraerán números adicionales en columnas separadas. Los números inválidos o ausentes se reemplazarán por `"NO DISPONIBLE"`.
7. **Corrección de nombres de personas:** Para los campos `SUPERVISOR` y `DIRECTOR`, se eliminarán tildes y espacios múltiples, y se corregirán inconsistencias en la escritura de nombres.
8. **Exportación final:** Una vez completado el proceso de limpieza, se exportará el dataset limpio a la siguiente ruta: `limpieza/output/datos_limpios.csv`.