# Proyecto 1 - Data Science
# Preparación y Limpieza de datos


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


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

Este cuaderno es la continuación del **Análisis de Datos** realizado anteriormente sobre el conjunto de datos obtenidos del sitio web del Mineduc, los cuales describen un grupo de centros de educación en Guatemala.

Cada paso esta acompañado de la referencia donde se analizó cierto aspecto del conjunto de datos.

## Preparación de Ambiente

**Unión e importación de datos**

Se realizó un script union_datos.py que une todos los datasets de los distintos departamentos en uno solo, esto para facilitar el manejo de datos.

In [70]:
```
import pandas as pd
import unicodedata
import sys
import re
import os

sys.path.append(os.path.abspath("../scripts"))
from union_datos import unir_dataframes

# Generar el DataFrame unificado
unir_dataframes()

# Importar el DataFrame generado
df = pd.read_csv("data_unificada.csv")
```

```
CSV combinado guardado en: .\data_unificada.csv
```

**Crear carpeta destino para alojar dataset limpio**

In [71]:
```
import os
import shutil

carpeta_destino = 'output'
carpeta_destino_csv = os.path.join(carpeta_destino, 'csv')

# Crear la carpeta si no existe
if not os.path.exists(carpeta_destino):
    os.makedirs(carpeta_destino)
    print(f"Carpeta '{carpeta_destino}' creada.")
```

**Duplicación de columnas para aplicarles la limpieza**

In [72]:
```
for col in df.columns:
    new_name = col + "_LIMPIO"
    df[new_name] = df[col]
```

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

In [73]:
```
for col in df.select_dtypes(include="object").columns:
    if col.endswith("_LIMPIO"):
        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.

# Limpieza de columnas

## DEPARTAMENTO

**Eliminar espacios inconsistentes y cualquier acentuación**

In [74]:
```
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_LIMPIO"] = df["DEPARTAMENTO_LIMPIO"].apply(limpiar_departamento)
```

## MUNICIPIO

**Eliminar espacios inconsistentes y cualquier acentuación**

In [75]:
```
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_LIMPIO"] = df["MUNICIPIO_LIMPIO"].apply(limpiar_municipio)
```

**Modificar los nombres de departamentos necesarios**

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 |

Por lo que se van a reemplazar dichos departamentos por el nombre completo para mantener información oficial valiosa.

In [76]:
```
correcciones_municipios = {
    "PETATAN": "SANTIAGO PETATAN",
    "LA TINTA": "SANTA CATALINA LA TINTA",
    "PACHALUN": "PACHALUM"
}

df["MUNICIPIO_LIMPIO"] = df["MUNICIPIO_LIMPIO"].replace(correcciones_municipios)
```

## ESTABLECIMIENTO

En el análisis se encontrraron nombres de algunos establecimientos que estaban escritos de manera distinta, por eso se busca estandarizar todos los nombres y así evitar variaciones.

In [77]:
```python
import re
import unicodedata
import pandas as pd

def estandarizar_establecimiento(nombre):
    if pd.isnull(nombre):
        return "NO REGISTRADO"

    # Quitar comillas, comas, guiones (incluso guiones largos Unicode)
    nombre = nombre.replace('"', '').replace("'", '')
    nombre = re.sub(r"[,\-–—]", " ", nombre)

    # Corregir NO. separados (NO. 2 → NO.2)
    nombre = re.sub(r"NO\.\s+(\d+)", r"NO.\1", nombre)

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

    # Eliminar tildes inversas, backticks y otros símbolos raros
    nombre = re.sub(r"[`\u0300-\u036f]", "", nombre)

    # Eliminar múltiples espacios y dejar uno solo
    nombre = re.sub(r"\s+", " ", nombre).strip()

    # Convertir a mayúsculas
    nombre = nombre.upper()

    # Correcciones específicas
    nombre = nombre.replace("EDCACION", "EDUCACION") \
                   .replace("NACIONA ", "NACIONAL ") \
                   .replace("COLEGO ", "COLEGIO ")

    # Eliminar punto final si existe
    nombre = re.sub(r"\.$", "", nombre)
    # ELiminar espacio al final si existe
    nombre = re.sub(r"\s+$","",nombre)

    return nombre

# Aplicar función al DataFrame
df["ESTABLECIMIENTO_LIMPIO"] = df["ESTABLECIMIENTO_LIMPIO"].apply(estandarizar_esta
```

Con los patrones observados, se realizó lo siguiente:

- Eliminar espacios extra entre palabras y ó números
- Eliminar acentuación y diéresis para estandarizar nombres
- Eliminar caractéres como `"`, `'`, `-`, entre otros
- Eliminar puntos y espacios al final
- Corregir faltas comunes de ortografía

## DIRECCIÓN

**Limpieza del campo y estandarización de abreviaturas**

In [78]:
```python
def limpiar_direccion(texto):
    if pd.isnull(texto):
        return "SIN INFORMACION"

    texto = str(texto).strip().upper()

    texto = unicodedata.normalize('NFKD', texto)
    texto = ''.join(c for c in texto if not unicodedata.combining(c))

    texto = texto.replace('"', '').replace("'", '').replace('-', '').replace('—', '
    texto = texto.replace(';', '').replace('.', '').replace(',', '')

    texto = re.sub(r'\bKM\.?(\d)', r'KM \1', texto)
```

```
    # Reemplazar abreviaturas comunes
    texto = re.sub(r'\bAV\b|\bAVENIDA\b|\bAVE\b', 'AVENIDA', texto)
    texto = re.sub(r'\bZ\b', 'ZONA', texto)
    texto = re.sub(r'\bCOL\b', 'COLONIA', texto)
    texto = re.sub(r'\s+', ' ', texto)

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

    return texto

df["DIRECCION_LIMPIA"] = df["DIRECCION"].apply(limpiar_direccion)
```

La columna `DIRECCION` representa una de las variables más susceptibles a inconsistencias en el formato, redacción y uso de caracteres especiales, ya que suele ser ingresada manualmente por distintas fuentes. Para garantizar la coherencia y la posibilidad de identificar direcciones duplicadas o similares, se aplicó un proceso de limpieza y estandarización conservador pero completo, implementado en una nueva columna llamada DIRECCION_LIMPIA.

## TELEFONO

**Realizar una limpieza básica de los números telefónicos, tratar valores nulos y estandarizarlos**

In [79]:
```
def limpiar_y_validar(numero):
    # Quitar todo lo que no sea número
    solo_numeros = re.sub(r"[^\d]", "", str(numero))

    if not solo_numeros:
        return None

    if len(solo_numeros) == 8:
        return solo_numeros

    # Si es más largo, asumimos que el número está al final
    if len(solo_numeros) > 8:
        posible = solo_numeros[-8:]
        return posible if len(posible) == 8 else None

    return None

def estandarizar_telefonos(valor):
    if pd.isna(valor) or str(valor).strip().upper() in {"NAN", "", "NO DISPONIBLE"}
        return "NO DISPONIBLE", "NO DISPONIBLE", "NO DISPONIBLE"

    partes = re.split(r"[\/,\-]", str(valor))

    telefonos_validos = []
    for parte in partes:
        numero = limpiar_y_validar(parte)
        if numero:
            telefonos_validos.append(numero)

    # Rellenar hasta 3
    telefonos_validos += ["NO DISPONIBLE"] * (3 - len(telefonos_validos))

    return tuple(telefonos_validos[:3])

# Aplicar la función y asignar dos columnas
df[["TELEFONO_LIMPIO", "TELEFONO_2_LIMPIO", "TELEFONO_3_LIMPIO"]] = df["TELEFONO"].
    lambda x: pd.Series(estandarizar_telefonos(x))
)
```

In [80]:
```
df["TELEFONOS_VALIDOS"] = df[["TELEFONO_LIMPIO", "TELEFONO_2_LIMPIO", "TELEFONO_3_L
    lambda row: sum(tel != "NO DISPONIBLE" for tel in row), axis=1
)

df["TELEFONOS_VALIDOS"].value_counts().sort_index()
```

Out[80]:
```
TELEFONOS_VALIDOS
0      63
1    6488
2      32
3       1
Name: count, dtype: int64
```

Se estandarizó el campo `TELEFONO` eliminando todos los caracteres no numéricos y normalizando los valores a un formato de 8 dígitos.
En caso de que un registro contuviera múltiples números separados por `/`, `,` o `-`, se extrajeron hasta tres números válidos y se asignaron a las columnas `TELEFONO_LIMPIO`, `TELEFONO_2_LIMPIO` y `TELEFONO_3_LIMPIO`.
Cuando un número tenía más de 8 dígitos, se conservaron los últimos 8 bajo el supuesto de que correspondían al número principal.
Si no se encontró ningún valor válido, se asignó `"NO DISPONIBLE"` en cada una de las columnas correspondientes.

## SUPERVISOR

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

In [81]:
```
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)
```

Reemplazar los nombres mal escritos por los valores correctos

In [82]:
```
reemplazos_supervisor = {
    "ROERICO DE LA CRUZ ICAL": "RODERICO DE LA CRUZ ICAL"
}

df["SUPERVISOR_LIMPIO"] = df["SUPERVISOR_LIMPIO"].replace(reemplazos_supervisor)
```

Se eliminaron tildes, se convirtió a mayúsculas, y se eliminaron espacios múltiples. Posteriormente se aplicó fuzzy matching para detectar nombres similares escritos de forma distinta (por ejemplo: "MARÍA LÓPEZ" vs "MARIA LOPEZ"). Se identificó el nombre de uno de los supervisores con errores tipográficos mediante fuzzy matching. Por ejemplo, ROERICO DE LA CRUZ ICAL tenía una similitud del 97.87% con RODERICO DE LA CRUZ ICAL. Por tanto, se estandarizó utilizando un diccionario de reemplazo.

## DIRECTOR

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

In [83]:
```
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)
```

Reemplazar los nombres mal escritos por su valor correcto

In [84]:
```
reemplazos_director = {
    "BRAYAN ONELL GAITAN RODRIGUEZ" : "BRYAN O'NELL GAITAN RODRIGUEZ",
    "MARTA NOREIGA" : "MARTA NORIEGA",
    "JULIO ENRIQUE JAREZ SILVESTRE" : "JULIO ENRIQUE JUAREZ SILVESTRE",
    "FREDY ANTONIO CUELLAR RODRUIGUEZ" : "FREDY ANTONIO CUELLAR RODRIGUEZ",
    "FRANCISCO CASTELLANOS MATUS" : "RAMON FRANCISCO CASTELLANOS MATUS",
    "MIGUEL ANGEL PISQUY QUIXTAN" : "MIGUEL ANGEL PISQUIY QUIXTAN",
    "VICTOR DENIEL VASQUEZ GOMEZ" : "VICTOR DANIEL VASQUEZ GOMEZ",
    "ARGELIA EDTIH MARROQUIN LOPEZ" : "ARGELIA EDITH MARROQUIN LOPEZ",
}

df["DIRECTOR_LIMPIO"] = df["DIRECTOR_LIMPIO"].replace(reemplazos_director)
```

Primero se estandarizaron los nombres eliminano los espacios dobles, eliminando las tildes y reemplazando posibles signos que podían causar problemas como las comillas( `"` ) entre nombres o incluso la comilla inversa `` ` ``.

Posteriormente se aplicó fuzzy matching para detectar nombres similares escritos de forma distinta (por ejemplo: "MARÍA LÓPEZ" vs "MARIA LOPEZ"). Se identificó el nombre de varios directores con errores tipográficos mediante fuzzy matching. Por ejemplo,MARTA NOREIGA tenía una similitud del 92.31% con MARTA NORIEGA.

Por tanto, se estandarizó utilizando un diccionario de reemplazo para cada director con el fin de mantener los nombres con una única forma de escribir.

# Exportación del Dataset limpio

Ahora que se ha realizado la limpieza y estandarización de los valores, se utilizarán las columnas procesadas (originalmente marcadas con el sufijo `_LIMPIO`) para generar una nueva versión del conjunto de datos.

Este nuevo archivo contendrá únicamente los campos ya preparados para análisis, y se exportará en la dirección `limpieza/output/datos_limpios.csv`.

In [85]:
```
# Seleccionar columnas limpias
columnas_limpias = [col for col in df.columns if col.endswith("_LIMPIO")]
df_limpio = df[columnas_limpias].copy()

# Renombrar columnas eliminando el sufijo _LIMPIO
df_limpio.columns = [col.replace("_LIMPIO", "") for col in columnas_limpias]

# Reordenar columnas de teléfono: TELEFONO, TELEFONO_2, TELEFONO_3
cols = list(df_limpio.columns)
tel_cols = ["TELEFONO_3","TELEFONO_2"]

for col in tel_cols:
    if col in cols:
        cols.remove(col)
        tel_index = cols.index("TELEFONO")
        cols.insert(tel_index + 1, col)

df_limpio = df_limpio[cols]

# Guardar CSV limpio
os.makedirs("../limpieza/output", exist_ok=True)
df_limpio.to_csv("../limpieza/output/datos_limpios.csv", index=False, encoding="utf
```

Así es como finalmente se concluye el proceso de limpieza, se estandarizaron todas las columnas del dataset original.

1. Se unificaron los datasets con el script union_datos.py.
2. Se duplicaron las columnas para aplicarles la limpieza
3. Se limpiaron las columnas de texto eliminando espacios extremos y convirtieron a mayúsculas Se verificó el `DEPARTAMENTO` y `MUNICIPIO` de cada centro educativo comparando los valores con datos oficiales extraídos del Instituto Nacional de Estadística Guatemala.
4. Para los nombres de los `ESTABLECIMIENTOS` se eliminaron caracteres especiales como `"`, `'`, `-`. Además de corregir faltas ortográficas en las palabras más comunes.
5. Para las `DIRECCIONES`, al ser una de las variables más susceptibles a inconsistencias en el formato, se aplicó un proceso de estandarización conservador. Se mantuvo un estándar para nombrar las calles o avenidas.
6. Para los `TELEFONOS` se encontraron varios valores con más de 8 dígitos, por lo que se aplicaron diversas reglas o condiciones para incluso agregar números extra para el caso de **17** y **26** dígitos. Además, los números inválidos se reemplazaron por `**NO DISPONIBLE**`.
7. Tanto para `SUPERVISOR` como para `DIRECTOR`, se estandarizaron los valores eliminando tildes y espacios múltiples. Además que, se corregiecon las inconsistencias de nombres mal escritos.
8. Finalmente se exporta el dataset limpio en la siguiente dirección: `limpieza/output/datos_limpios.csv`.