DATA CLEANING AUTOMATION AND MERGING

1 Cleaning Automation of VAERS Databases (2020-2025)

```
1.  # Author: Miguel Atarés
2.  # coding: utf-8

3.  import pandas as pd

4.  def limpiar_vaers_data(path_csv: str, anio: int, guardar_csv=False,
    ruta_guardado=None):
5.  """
6.  Limpia un archivo VAERS DATA según los pasos realizados en el análisis de 2021.

7.  Parámetros:
        - path_csv: ruta del archivo CSV original (string)
        - anio: año del archivo (por ahora no afecta, pero puede servir para
          adaptar filtros en el futuro)
        - guardar_csv: True si quieres guardar el archivo limpio en disco
        - ruta_guardado: ruta donde guardar el archivo limpio (si guardar_csv es
          True)

8.  Devuelve:
        - DataFrame limpio (pandas)
9.  """

10. # Leer el CSV
11. df = pd.read_csv(path_csv, encoding="ISO-8859-1", delimiter=",",
    low_memory=False)

12. print("\n📊 Información INICIAL del dataset:")
13. print(f"Forma (filas, columnas): {df.shape}")
14. df.info()
15. print("\nDescribe (solo columnas numéricas):")
16. print(df.describe())
17. print("\nPorcentaje de nulos por columna:")
18. print(df.isnull().mean().sort_values(ascending=False) * 100)

19. # Columnas que deben mantenerse
20. columnas_a_conservar = [
        - "VAERS_ID", "RECVDATE", "VAX_DATE", "ONSET_DATE", "NUMDAYS",
          "DATEDIED",
        - "STATE", "AGE_YRS", "SEX",
        - "DIED", "L_THREAT", "ER_VISIT", "HOSPITAL", "HOSPDAYS",
        - "DISABLE", "RECOVD", "BIRTH_DEFECT"
21. ]
```

```
columnas_presentes = set(df.columns)
faltantes = set(columnas_a_conservar) - columnas_presentes
if faltantes:
    - raise ValueError(f"FALTAN columnas obligatorias en el archivo:
      {faltantes}")

df = df[columnas_a_conservar].copy()

df.drop_duplicates(inplace=True)

fechas = ["RECVDATE", "VAX_DATE", "ONSET_DATE", "DATEDIED"]
for col in fechas:
    - df[col] = pd.to_datetime(df[col], errors="coerce")

# Reemplazar valores problemáticos
df["RECOVD"] = df["RECOVD"].replace("U", pd.NA)
df["STATE"] = df["STATE"].str.upper()

estados_validos = [
    - 'AL','AK','AZ','AR','CA','CO','CT','DE','FL','GA','HI','ID','IL','IN','
      IA','KS','KY','LA','ME','MD','MA','MI',
    - 'MN','MS','MO','MT','NE','NV','NH','NJ','NM','NY','NC','ND','OH','OK','
      OR','PA','RI','SC','SD','TN','TX','UT',
    - 'VT','VA','WA','WV','WI','WY','DC','PR','VI','GU','AS','MP','MH','FM'
]
df.loc[~df["STATE"].isin(estados_validos), "STATE"] = pd.NA
df.dropna(subset=["STATE"], inplace=True)

columnas_binarias = ["DIED", "L_THREAT", "HOSPITAL", "DISABLE", "BIRTH_DEFECT",
"ER_VISIT"]
for col in columnas_binarias:
    - df[col] = df[col].fillna("N")

# Fechas válidas (ajustables)
fecha_min = pd.to_datetime(f"{anio}-01-01")
fecha_max = pd.to_datetime(f"{anio + 1}-01-15")
df = df[
    - (df["VAX_DATE"].isna() | ((df["VAX_DATE"] >= fecha_min) &
      (df["VAX_DATE"] <= fecha_max))) &
    - (df["ONSET_DATE"].isna() | ((df["ONSET_DATE"] >= fecha_min) &
      (df["ONSET_DATE"] <= fecha_max))) &
    - (df["DATEDIED"].isna() | ((df["DATEDIED"] >= fecha_min) &
      (df["DATEDIED"] <= fecha_max)))
]

# HOSPDAYS máximo de 1 año
df = df[(df["HOSPDAYS"].isna()) | (df["HOSPDAYS"] <= 365)]
```

```
print("\n📊 Información FINAL del dataset:")
print(f"Forma (filas, columnas): {df.shape}")
df.info()
print("\nDescribe (solo columnas numéricas):")
print(df.describe())
print("\nPorcentaje de nulos por columna:")
print(df.isnull().mean().sort_values(ascending=False) * 100)

# Guardar si se solicita
if guardar_csv and ruta_guardado:
    - df.to_csv(ruta_guardado, index=False)
    - print(f"✅ Archivo limpio guardado en: {ruta_guardado}")

return df

df_2022_Data = limpiar_vaers_data(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2022VAERSData\2022VAERSDATA.csv",
anio=2022,
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2022Data_limpio_para_powerbi.csv"
)

def limpiar_vaers_symptoms(path_csv: str, guardar_csv=False, ruta_guardado=None):
"""
Limpia un archivo VAERS SYMPTOMS eliminando columnas irrelevantes y duplicados.

Parámetros:
    - path_csv: ruta del archivo original
    - guardar_csv: si True, guarda el resultado
    - ruta_guardado: ruta donde guardar el archivo limpio

Devuelve:
    - DataFrame limpio
"""

# Leer el CSV original
df = pd.read_csv(path_csv, encoding="ISO-8859-1", delimiter=",")

# 📊 Información inicial
print("\n📊 Información INICIAL del dataset:")
print(f"Forma (filas, columnas): {df.shape}")
df.info()
print("\nDescribe (solo columnas numéricas):")
print(df.describe())
print("\nPorcentaje de nulos por columna:")
print(df.isnull().mean().sort_values(ascending=False) * 100)
```

```
# Columnas a eliminar
columnas_a_eliminar = [
    - "SYMPTOMVERSION1", "SYMPTOMVERSION2", "SYMPTOMVERSION3",
    - "SYMPTOMVERSION4", "SYMPTOMVERSION5", "ORDER"
]
df = df.drop(columns=[col for col in columnas_a_eliminar if col in df.columns])

# Eliminar duplicados
df.drop_duplicates(inplace=True)

# 📊 Información final
print("\n📊 Información FINAL del dataset:")
print(f"Forma (filas, columnas): {df.shape}")
df.info()
print("\nDescribe (solo columnas numéricas):")
print(df.describe())
print("\nPorcentaje de nulos por columna:")
print(df.isnull().mean().sort_values(ascending=False) * 100)

# Guardar si se solicita
if guardar_csv and ruta_guardado:
    - df.to_csv(ruta_guardado, index=False)
    - print(f"\n✅ Archivo limpio guardado en: {ruta_guardado}")

return df

df_2022_Symptoms = limpiar_vaers_symptoms(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2022VAERSData\2022VAERSSYMPTOMS.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2022Symptoms_limpio_para_powerbi.csv"
)

def limpiar_vaers_vax(path_csv: str, guardar_csv=False, ruta_guardado=None):
"""
Limpia un archivo VAERS VAX eliminando columnas irrelevantes,
filtrando vacunas COVID y mostrando análisis de 'UNK' en dosis.

Parámetros:
    - path_csv: ruta del archivo original
    - guardar_csv: si True, guarda el resultado
    - ruta_guardado: ruta donde guardar el archivo limpio

Devuelve:
    - DataFrame limpio
"""
```

```
# Leer CSV
df = pd.read_csv(path_csv, encoding="ISO-8859-1", delimiter=",")

# 📊 Información inicial
print("\n📊 Información INICIAL del dataset:")
print(f"Forma (filas, columnas): {df.shape}")
df.info()
print("\nDescribe (solo columnas numéricas):")
print(df.describe())
print("\nPorcentaje de nulos por columna:")
print(df.isnull().mean().sort_values(ascending=False) * 100)

# Columnas a eliminar
columnas_a_eliminar = ["VAX_LOT", "VAX_ROUTE", "VAX_SITE", "VAX_NAME", "ORDER"]
df = df.drop(columns=[col for col in columnas_a_eliminar if col in df.columns])

# Filtrar solo vacunas COVID19
df = df[df["VAX_TYPE"].str.contains("COVID19", case=False, na=False)]

# Eliminar columna redundante VAX_TYPE
df = df.drop(columns=["VAX_TYPE"])

# Eliminar duplicados
df.drop_duplicates(inplace=True)

# 📊 Información final
print("\n📊 Información FINAL del dataset:")
print(f"Forma (filas, columnas): {df.shape}")
df.info()
print("\nDescribe (solo columnas numéricas):")
print(df.describe())
print("\nPorcentaje de nulos por columna:")
print(df.isnull().mean().sort_values(ascending=False) * 100)

# Porcentaje de UNK en VAX_DOSE_SERIES
if "VAX_DOSE_SERIES" in df.columns:
        - total = len(df)
        - unk_count = (df["VAX_DOSE_SERIES"] == "UNK").sum()
        - unk_pct = round((unk_count / total) * 100, 2)
        - print(f"\n🔎 'UNK' en VAX_DOSE_SERIES: {unk_count} filas
          ({unk_pct}%)")

# Guardar si se solicita
if guardar_csv and ruta_guardado:
        - df.to_csv(ruta_guardado, index=False)
        - print(f"\n✅ Archivo limpio guardado en: {ruta_guardado}")

return df
```

```
df_vax_2022 = limpiar_vaers_vax(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2022VAERSData\2022VAERSVAX.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2022VAX_limpio_para_powerbi.csv"
)
# Ahora que ya hemos hecho la prueba, ejecutamos loos scripts de limpieza para el resto de años
df_2023_Data = limpiar_vaers_data(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2023VAERSData\2023VAERSDATA.csv",
anio=2023,
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2023Data_limpio_para_powerbi.csv"
)

df_2024_Data = limpiar_vaers_data(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2024VAERSData\2024VAERSDATA.csv",
anio=2024,
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2024Data_limpio_para_powerbi.csv"
)

df_2025_Data = limpiar_vaers_data(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2025VAERSData\2025VAERSDATA.csv",
anio=2025,
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2025Data_limpio_para_powerbi.csv"
)

df_2020_Data = limpiar_vaers_data(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2020VAERSData\2020VAERSDATA.csv",
anio=2020,
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2020Data_limpio_para_powerbi.csv"
)

df_2023_Symptoms = limpiar_vaers_symptoms(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2023VAERSData\2023VAERSSYMPTOMS.csv",
guardar_csv=True,
```

```
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2023Symptoms_limpio_para_powerbi.csv"
)

df_2024_Symptoms = limpiar_vaers_symptoms(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2024VAERSData\2024VAERSSYMPTOMS.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2024Symptoms_limpio_para_powerbi.csv"
)

df_2025_Symptoms = limpiar_vaers_symptoms(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2025VAERSData\2025VAERSSYMPTOMS.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2025Symptoms_limpio_para_powerbi.csv"
)

df_2020_Symptoms = limpiar_vaers_symptoms(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2020VAERSData\2020VAERSSYMPTOMS.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2020Symptoms_limpio_para_powerbi.csv"
)

df_vax_2023 = limpiar_vaers_vax(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2023VAERSData\2023VAERSVAX.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2023VAX_limpio_para_powerbi.csv"
)

df_vax_2024 = limpiar_vaers_vax(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2024VAERSData\2024VAERSVAX.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2024VAX_limpio_para_powerbi.csv"
)

df_vax_2025 = limpiar_vaers_vax(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2025VAERSData\2025VAERSVAX.csv",
guardar_csv=True,
```

```
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2025VAX_limpio_para_powerbi.csv"
)

df_vax_2020 = limpiar_vaers_vax(
path_csv=r"C:\Users\migue\Desktop\COVID-19 Project\Databases\2020VAERSData\2020VAERSVAX.csv",
guardar_csv=True,
ruta_guardado=r"C:\Users\migue\Desktop\COVID-19 Project\dfVAERS2020VAX_limpio_para_powerbi.csv"
)
```

2 Merge of all VAERS Databases (2020-2025)

```
# Author: Miguel Atarés
# coding: utf-8

import pandas as pd

# Leemos los archivos individualmente
df_2020 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2020\dfVAERS2020Data_limpio_para_powerbi.csv")
df_2020["YEAR"] = 2020

df_2021 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2021\dfVAERS2021Data_limpio_para_powerbi.csv")
df_2021["YEAR"] = 2021

df_2022 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2022\dfVAERS2022Data_limpio_para_powerbi.csv")
df_2022["YEAR"] = 2022

df_2023 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2023\dfVAERS2023Data_limpio_para_powerbi.csv")
df_2023["YEAR"] = 2023

df_2024 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2024\dfVAERS2024Data_limpio_para_powerbi.csv")
df_2024["YEAR"] = 2024

df_2025 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2025\dfVAERS2025Data_limpio_para_powerbi.csv")
df_2025["YEAR"] = 2025

# Unimos todos
df_merged = pd.concat([df_2020, df_2021, df_2022, df_2023, df_2024, df_2025], ignore_index=True)
```

```
# Guardamos el resultado
df_merged.to_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VAERS_Data_2020_2025.csv", index=False)

print("✅ Merge completado para VAERS Data.")

# Leer archivos SYMPTOMS
df_symptoms_2020 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2020\dfVAERS2020Symptoms_limpio_para_powerbi.csv")
df_symptoms_2020["YEAR"] = 2020

df_symptoms_2021 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2021\dfVAERS2021Symptoms_limpio_para_powerbi.csv")
df_symptoms_2021["YEAR"] = 2021

df_symptoms_2022 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2022\dfVAERS2022Symptoms_limpio_para_powerbi.csv")
df_symptoms_2022["YEAR"] = 2022

df_symptoms_2023 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2023\dfVAERS2023Symptoms_limpio_para_powerbi.csv")
df_symptoms_2023["YEAR"] = 2023

df_symptoms_2024 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2024\dfVAERS2024Symptoms_limpio_para_powerbi.csv")
df_symptoms_2024["YEAR"] = 2024

df_symptoms_2025 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2025\dfVAERS2025Symptoms_limpio_para_powerbi.csv")
df_symptoms_2025["YEAR"] = 2025

# Concatenar
df_symptoms_merged = pd.concat([
df_symptoms_2020, df_symptoms_2021, df_symptoms_2022,
df_symptoms_2023, df_symptoms_2024, df_symptoms_2025
], ignore_index=True)

# Guardar
df_symptoms_merged.to_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VAERS_Symptoms_2020_2025.csv", index=False)

print("✅ Merge completado para VAERS Symptoms.")
```

```python
# Leer archivos VAX
df_vax_2020 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2020\dfVAERS2020VAX_limpio_para_powerbi.csv")
df_vax_2020["YEAR"] = 2020

df_vax_2021 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2021\dfVAERS2021VAX_limpio_para_powerbi.csv")
df_vax_2021["YEAR"] = 2021

df_vax_2022 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2022\dfVAERS2022VAX_limpio_para_powerbi.csv")
df_vax_2022["YEAR"] = 2022

df_vax_2023 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2023\dfVAERS2023VAX_limpio_para_powerbi.csv")
df_vax_2023["YEAR"] = 2023

df_vax_2024 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2024\dfVAERS2024VAX_limpio_para_powerbi.csv")
df_vax_2024["YEAR"] = 2024

df_vax_2025 = pd.read_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VARS\2025\dfVAERS2025VAX_limpio_para_powerbi.csv")
df_vax_2025["YEAR"] = 2025

# Concatenar
df_vax_merged = pd.concat([
df_vax_2020, df_vax_2021, df_vax_2022,
df_vax_2023, df_vax_2024, df_vax_2025
], ignore_index=True)

# Guardar
df_vax_merged.to_csv(r"G:\Mi unidad\PROGRAMACIÓN\Portfolio Projetcs\1. COVID-19\Databases Limpias\VAERS_VAX_2020_2025.csv", index=False)

print("✅ Merge completado para VAERS VAX.")

# Calcular totales individuales
total_vax = len(df_vax_2020) + len(df_vax_2021) + len(df_vax_2022) + len(df_vax_2023) + len(df_vax_2024) + len(df_vax_2025)
total_symptoms = len(df_symptoms_2020) + len(df_symptoms_2021) + len(df_symptoms_2022) + len(df_symptoms_2023) + len(df_symptoms_2024) + len(df_symptoms_2025)
total_data = len(df_2020) + len(df_2021) + len(df_2022) + len(df_2023) + len(df_2024) + len(df_2025)

# Prints comparativos
print("VAX → Individual:", total_vax, "| Mergeado:", len(df_vax_merged), "|", "✅ OK" if total_vax == len(df_vax_merged) else "❌ Mismatch")
```

```
70. print("SYMPTOMS → Individual:", total_symptoms, "| Mergeado:",
    len(df_symptoms_merged), "|", "✅ OK" if total_symptoms ==
    len(df_symptoms_merged) else "❌ Mismatch")
71. print("DATA → Individual:", total_data, "| Mergeado:", len(df_merged), "|", "✅
    OK" if total_data == len(df_merged) else "❌ Mismatch")
```