PYTHON DATA CLEANING SCRIPTS

1 Cleaning of OWID Database *compact.csv*

```
# ETL Cleaning Script for OWID Datasets (compact.csv and vaccinations_manufacturer.csv)
# Author: Miguel Atarés
# This script performs data cleaning and preprocessing on OWID datasets using Python and pandas.

import pandas as pd

df = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19 Project\Databases\compact.csv', encoding = "ISO-8859-1", delimiter=',')
df.head(5)
df.shape
df.describe()
df.info()
# Lo primero es eliminar las columnas que ya he planeado que no quiero
df2 = df.drop(['code', 'new_cases', 'new_cases_per_million', 'new_deaths_smoothed',
'new_deaths_smoothed_per_million','hosp_patients','weekly_hosp_admissions',
'weekly_hosp_admissions', 'icu_patients_per_million', 'weekly_icu_admissions',
'total_tests', 'new_tests', 'new_tests_per_thousand', 'total_vaccinations',
'people_vaccinated', 'people_fully_vaccinated', 'new_vaccinations',
'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
'total_boosters_per_hundred', 'new_vaccinations_smoothed_per_million',
'new_people_vaccinated_smoothed', 'new_people_vaccinated_smoothed_per_hundred'],
axis = 1)
df2.info() #Todos los valores numéricos están como floats, no hay errores de columnas en string, solo tipo date que no es datetime
df2.isnull().mean().sort_values(ascending=False) #Me da % de valoroes nulos. A más alto, más valores nulos

# Vemos que hay 2 columnas con 100% de nan y es un valor que se puede buscar y poner manualmente, pero otros tienen % alto de nans no se porque aun
df2.shape
# Vamos a eliminar duplicados:
df2.drop_duplicates(inplace=True)
df2.shape #Vemos que no hay duplicados
lista_paises = df2["country"].unique()
lista_paises
len(lista_paises)
# Muchos "paises" no son ni paises, y otros me entorpecen el analisis pues se detectan muchos nan al ser países poco desarrollados y no reportar datos
```

```
# Vamos a eliminar los paises que no nos interesan, ya que no tienen datos suficientes, quedandome con los paises mas desarrollados y con más datos

latam_top10 = [
'Argentina', 'Brazil', 'Chile', 'Colombia', 'Mexico', 'Peru',
'Uruguay', 'Costa Rica', 'Ecuador', 'Panama'
]

africa_top10 = [
'Nigeria', 'South Africa', 'Egypt', 'Morocco', 'Algeria',
'Kenya', 'Ethiopia', 'Ghana', 'Ivory Coast', 'Tunisia'
]

asia_top10 = [
'China', 'India', 'Japan', 'South Korea', 'Indonesia', 'Iran',
'Saudi Arabia', 'Israel', 'Vietnam', 'Thailand'
]

norteamerica = [
'Canada', 'United States'
]

europa = [
'United Kingdom', 'Germany', 'France', 'Italy', 'Spain', 'Netherlands',
'Belgium', 'Sweden', 'Switzerland', 'Poland', 'Portugal', 'Austria',
'Ireland', 'Norway', 'Finland', 'Denmark', 'Czechia', 'Hungary',
'Greece', 'Slovakia', 'Slovenia', 'Romania', 'Bulgaria', 'Croatia',
'Estonia', 'Lithuania', 'Latvia', 'Luxembourg', 'Malta'
]

oceania = [
'Australia', 'New Zealand', 'Fiji', 'Papua New Guinea'
]

# Unimos todos los países a conservar en una sola lista
paises_validos = set(latam_top10 + africa_top10 + asia_top10 + norteamerica + europa + oceania)


df3 = df2[df2['country'].isin(paises_validos)].copy() #Con esto creamos un nuevo df con los paises interesantes
df3.shape #Hemos reducido muchísimo el dataset, de 506841 a 128002 filas.
df3['country'].unique() #Comprobamos que todo este ok.
len(df3['country'].unique()) #Ahora solo tenemos 64 paises, y columnas reducidas claro.
# Ahora vamos a ver cuantos nan hay en total y comparamos
df3.isnull().mean().sort_values(ascending=False) #Sigue habiendo muchas columnas con alto % de NaNs.
```

```
# Vamos a ver que paises no reportan datos en ninguna columna para ver si los podemos descartar
# Lista de columnas con alto porcentaje de nan (excluyendo las totalmente vacías)
cols_a_comprobar = [
"excess_mortality",
"excess_mortality_cumulative",
"excess_mortality_cumulative_absolute",
"excess_mortality_cumulative_per_million",
"weekly_icu_admissions_per_million",
"weekly_hosp_admissions_per_million",
"total_boosters",
"hosp_patients_per_million",
"icu_patients",
"people_fully_vaccinated_per_hundred",
"handwashing_facilities",
"total_tests_per_thousand",
"tests_per_case",
"new_tests_smoothed_per_thousand",
"new_tests_smoothed",
"positive_rate",
"new_vaccinations_smoothed",
"reproduction_rate",
"stringency_index"
]

# Iteramos por cada columna y mostramos los países que tienen todo nan en esa columna
for col in cols_a_comprobar:
paises_nunca_reportan = df3.groupby("country")[col].apply(lambda x: x.isna().all())
paises_nunca_reportan = paises_nunca_reportan[paises_nunca_reportan].index.tolist()
print(f"\n📉 Países que NUNCA reportan datos en: {col}")
print(paises_nunca_reportan if paises_nunca_reportan else "Todos los países tienen al menos algún dato.")

# Con esto ya sabemos paises con muchos huescos en los datos. sin embargo, decido no eliminar más países. lo tendré en cuenta para la visualización en Power BI
df3["date"].sort_values(ascending=False) #Revisamos fechas y veo que hay fechas que ni si quiera han ocurrido aun... Vamos a eliminarlas
# Primero, convertimos la fecha a datetime64[ns] de pandas:
# Convertir la columna 'date' a tipo datetime
df3["date"] = pd.to_datetime(df3["date"], errors='coerce')

# Eliminar filas con fecha posterior al 1 de junio de 2025
df3 = df3[df3["date"] <= "2025-06-01"]
```

```
df3.shape
# Reducimos algo, pero menos de lo esperado. de 128002 a 126656

# Lo que vamos a hacer ahora es rellenar 3 columnas que podemos obtener con datos oficiales en internet y más o menos se mantienen establesa lor largo del tiempo
# El resto de columnas con alto % de nan las dejaremos así
# Las columnas que podemos rellenar manualmente son: human_development_index, life_expectancy y handwashing_facilities
df3.groupby('country')['handwashing_facilities'].mean().reset_index() #Miramos con groupby el handwshing facilities por country antes de meterlo
# Diccionarios con tus valores recopilados
idh = {
"Algeria": 0.745, "Argentina": 0.842, "Australia": 0.951, "Austria": 0.916, "Belgium": 0.937,
"Brazil": 0.754, "Bulgaria": 0.795, "Canada": 0.936, "Chile": 0.855, "China": 0.768,
"Colombia": 0.752, "Costa Rica": 0.809, "Croatia": 0.858, "Czechia": 0.889, "Denmark": 0.948,
"Ecuador": 0.740, "Egypt": 0.731, "Estonia": 0.890, "Ethiopia": 0.498, "Fiji": 0.730,
"Finland": 0.940, "France": 0.903, "Germany": 0.942, "Ghana": 0.632, "Greece": 0.887,
"Hungary": 0.846, "India": 0.633, "Indonesia": 0.705, "Iran": 0.774, "Ireland": 0.945,
"Israel": 0.919, "Italy": 0.895, "Japan": 0.925, "Kenya": 0.575, "Latvia": 0.863,
"Lithuania": 0.875, "Luxembourg": 0.930, "Malta": 0.918, "Mexico": 0.758, "Morocco": 0.683,
"Netherlands": 0.941, "New Zealand": 0.937, "Nigeria": 0.535, "Norway": 0.961, "Panama": 0.805,
"Papua New Guinea": 0.558, "Peru": 0.762, "Poland": 0.876, "Portugal": 0.866, "Romania": 0.821,
"Saudi Arabia": 0.875, "Slovakia": 0.848, "Slovenia": 0.918, "South Africa": 0.713,
"South Korea": 0.925, "Spain": 0.905, "Sweden": 0.947, "Switzerland": 0.962, "Thailand": 0.800,
"Tunisia": 0.731, "United Kingdom": 0.929, "United States": 0.921, "Uruguay": 0.809,
"Vietnam": 0.703
}

vida = {
"Algeria": 76.4, "Argentina": 75.4, "Australia": 84.5, "Austria": 81.6, "Belgium": 81.9,
"Brazil": 72.8, "Bulgaria": 71.8, "Canada": 82.7, "Chile": 78.9, "China": 78.2,
"Colombia": 72.8, "Costa Rica": 77.0, "Croatia": 77.6, "Czechia": 77.7, "Denmark": 81.4,
"Ecuador": 73.7, "Egypt": 70.2, "Estonia": 77.1, "Ethiopia": 65.0, "Fiji": 67.1,
```

```
"Finland": 82.0, "France": 82.5, "Germany": 80.6, "Ghana": 63.8, "Greece": 80.1,
"Hungary": 74.5, "India": 67.2, "Indonesia": 67.6, "Iran": 73.9, "Ireland": 82.0,
"Israel": 82.3, "Italy": 82.9, "Japan": 84.8, "Kenya": 61.4, "Latvia": 73.6,
"Lithuania": 73.7, "Luxembourg": 82.6, "Malta": 83.8, "Mexico": 70.2, "Morocco": 74.0,
"Netherlands": 81.7, "New Zealand": 82.5, "Nigeria": 52.7, "Norway": 83.2, "Panama": 76.2,
"Papua New Guinea": 65.4, "Peru": 72.4, "Poland": 76.5, "Portugal": 81.0, "Romania": 74.2,
"Saudi Arabia": 76.9, "Slovakia": 74.9, "Slovenia": 80.7, "South Africa": 62.3,
"South Korea": 83.7, "Spain": 83.0, "Sweden": 83.0, "Switzerland": 84.0, "Thailand": 78.7,
"Tunisia": 73.8, "United Kingdom": 80.7, "United States": 77.2, "Uruguay": 75.4,
"Vietnam": 73.6
}

manos = {
"Algeria": 0.85, "Argentina": 0.95, "Australia": 0.99, "Austria": 0.99, "Belgium": 0.99,
"Brazil": 0.86, "Bulgaria": 0.95, "Canada": 0.99, "Chile": 0.70, "China": 0.90,
"Colombia": 0.70, "Costa Rica": 0.86, "Croatia": 0.90, "Czechia": 0.99, "Denmark": 0.94,
"Ecuador": 0.87, "Egypt": 0.90, "Estonia": 0.99, "Ethiopia": 0.08, "Fiji": 0.95,
"Finland": 0.99, "France": 0.99, "Germany": 0.99, "Ghana": 0.42, "Greece": 0.99,
"Hungary": 0.90, "India": 0.38, "Indonesia": 0.79, "Iran": 0.95, "Ireland": 0.99,
"Israel": 0.99, "Italy": 0.99, "Japan": 0.99, "Kenya": 0.38, "Latvia": 0.99,
"Lithuania": 0.99, "Luxembourg": 0.99, "Malta": 0.92, "Mexico": 0.86, "Morocco": 0.84,
"Netherlands": 0.99, "New Zealand": 0.99, "Nigeria": 0.37, "Norway": 0.94, "Panama": 0.87,
"Papua New Guinea": 0.13, "Peru": 0.70, "Poland": 0.99, "Portugal": 0.90, "Romania": 0.95,
"Saudi Arabia": 0.97, "Slovakia": 0.99, "Slovenia": 0.90, "South Africa": 0.44,
"South Korea": 0.95, "Spain": 1.00, "Sweden": 0.99, "Switzerland": 0.99, "Thailand": 0.39,
"Tunisia": 0.84, "United Kingdom": 1.00, "United States": 1.00, "Uruguay": 0.98,
"Vietnam": 0.40
}
# Aplicar los valores manuales al dataframe. con .map(diccionario)
df3["human_development_index"] = df3["country"].map(idh)
df3["life_expectancy"] = df3["country"].map(vida)
df3["handwashing_facilities"] = df3["country"].map(manos)
# Revisamos a ver si es corecto
df3.groupby("country")[["human_development_index", "life_expectancy", "handwashing_facilities"]].mean().reset_index()

df3.isnull().mean().sort_values(ascending=False)
```

```
# Vamos a ver porque hay tantos nan en algunas variables como mortalidad. Parece que columnas como las de excess_mortality reportan solo 1 vez al mes, por ello hay mucho % de NaNs
resultado = df3.groupby('country')['excess_mortality'].mean().reset_index()

# Mostrar todas las filas una por una
for index, row in resultado.iterrows():
print(row)

# Me da media casi todos los valores, eso significa que en este caso, no es porque ciertos paises nunca reporten, si no que algunos paises han debido reportar en algun momento, pero pocas veces
# Se me ha olvidado quitar valores anteriores a la pandemia. Datos empiezan 01/01/2020 y se declaro en 11/03/2020, por lo que establecemos eliminar desde el 01/03/2020
df3 = df3[df3["date"] >= "2020-03-01"]
df3.shape #Hemos reducido de 126656 a 122816, perfecto.
# Por último, reviso columnas categóricas
resultado = df3.groupby("country")["continent"].nunique().sort_values(ascending=False)

for country, num_continentes in resultado.items():
print(f"{country}: {num_continentes}")
# ¿Cada pais tiene correctamente asociado su contienente? Lo revisamos con este código:
agrupado = df3.groupby("country")["continent"].unique()

for country, continents in agrupado.items():
print(f"{country}: {list(continents)}")

# Perfecto, todas dan un solo valor de continente y he revisado que esté bien, por lo que todo ok
df3.to_csv("df3_limpio_para_powerbi.csv", index=False)
# Vale, me faltaba hacer un describe() para revisar valores que tiene cada columna ya que muchos pueden ser ilógicos
pd.set_option('display.max_columns', None)

df3.describe()
# 1. tests_per_case:
# Problema: la mediana es ≈17, pero el máximo es 789.603, lo cual sugiere que por cada positivo hubo casi 790.000 test... absurdo
# Umbral lógico: más de 500 tests por caso se considera probablemente erróneo
# 2. reproduction_rate:
# Problema: mínimo negativo (-0.056) y máximo 4.64. el valor negativo no tiene sentido epidemiológicamente
# Umbral lógico: no debería ser menor que 0

# 3. otras columnas (no limpiaremos más)
```

```
# Por ejemplo, new_cases_smoothed de 5 millones en un dia, ratios de excess mortality muy elevados o negativos, etc. hay muchos valores de estos
# Que considerados sospechosos. pero haciendo un poco de research, sabemos que hay días en loos que el covid pegó muy fuerte, por lo que puede
# Ser que haya habido registros así en ciertas fechas, al final una pandemia puede escalar exponencialmente en ciertos momentos
# Vamos a definir condiciones lógicas para seguir limipiando los datos y ver si hay errores en los valores extremos
errores = {
"tests_per_case": df3["tests_per_case"] > 500,
"reproduction_rate": df3["reproduction_rate"] < 0,
}

# Ver cuántas filas están afectadas por cada condición
for col, condition in errores.items():
print(f"{col}: {condition.sum()} filas afectadas")

# Ver ejemplos de los valores problemáticos
for col, condition in errores.items():
print(f"\nEjemplos de errores en {col}:")
display(df3[condition][["country", "date", col]].head(5))

# Crear copia limpia y reemplazar los valores extremos por nan
df4 = df3.copy()
for col, condition in errores.items():
df4.loc[condition, col] = pd.NA

# Verificación final de limpieza aplicada
print("Recuento tras limpieza:")
print("Reproduction rate negativos:", df4["reproduction_rate"].lt(0).sum())
print("Tests per case mayores a 500:", df4["tests_per_case"].gt(500).sum())

df4.describe() #Comprobamos que este todo ok conforme lo que queriamos.
# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en power bi con este dataset. Repasemos lo hecho o no hecho aquí:
# ✅ 2.1) limpiar los nan
# ✅ 2.2) formato datetime
# ⏳ 2.3) modificar decimales --> en pbi
# ✅ 2.4) revisar duplicados
# ✅ 2.5) revisar columnas categóricas
# ✅ 2.6) outliers (iqr, boxplots) --> outliers y medidas extrañas comprobados con describe()
# ❌ 2.7) distribución de datos (histogramas, frecuencias) --> en pbi
# ✅ 2.8) filtrar fechas previas a la pandemia
# ❌ 2.9) correlaciones --> realizaremos en power bi
# ✅ 2.10) eliminar columnas innecesarias
# ✅ 2.11) revisar valores de desviacion como minimos y maximos de columnas

df4.to_csv("df4_limpio_para_powerbi.csv", index=False)
```

```
# Ahora empezamos la limpieza de otra base de datos: vaccinations manufacturer
# Solo hay que poner los mismos paises, controlar las mismas fechas y revisar columna de tipos de vacuna
df_vacc = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19 Project\Databases\vaccinations_manufacturer.csv', encoding = "ISO-8859-1", delimiter=',')
df_vacc.shape
df_vacc.info()
df_vacc2 = df_vacc[df_vacc['country'].isin(paises_validos)].copy()
df_vacc2.shape
df_vacc2['country'].unique()
# Esto significa que faltan algunos países en el dataset, simplemente porque no habría datos de muchos de ellos
# Además el número de filas no ha bajado mucho, por lo que originalmente en el dataset había pocos países, muchos ya incluidos en nuestra lsita

# Esta es la lista de paises que nos faltan en este dataset y están en el primero:
# Algeria, australia, brazil, china, colombia, costa rica, egypt, ethiopia, fiji, ghana, greece, india, indonesia, iran, israel, kenya, mexico,
# Morocco, new zealand, nigeria, panama, papua new guinea, saudi arabia, thailand, tunisia, united kingdom, vietnam

# Ahora transformamos la fecha y excluimos las que no nos interesan:
# Convertir la columna 'date' a tipo datetime
df_vacc2["date"] = pd.to_datetime(df_vacc2["date"], errors='coerce')

# Filtrar filas solo entre el 1 de marzo de 2020 y el 1 de junio de 2025 (ambos inclusive)
df_vacc2 = df_vacc2[(df_vacc2["date"] >= "2020-03-01") & (df_vacc2["date"] <= "2025-06-01")]

df_vacc2.info()
df_vacc2.shape
df_vacc2['vaccine'].unique() #Vemos como está categorizado la columna vaccine
# Eliminamos los duplicados
df_vacc2.drop_duplicates(inplace=True)
df_vacc2.shape #No hay duplicados
# Me interesa crear una nueva columna para poder asociar cada nombre de vaccine con el tipo de tecnología que usa la propia vacuna
# Diccionario de mapeo vacuna → tecnología
vaccine_technology = {
"Pfizer/BioNTech": "ARNm",
"Moderna": "ARNm",
"Oxford/AstraZeneca": "Vector viral",
"Johnson&Johnson": "Vector viral",
"Sputnik V": "Vector viral",
```

```
253. "Sputnik Light": "Vector viral",
254. "CanSino": "Vector viral",
255. "Sinovac": "Virus inactivado",
256. "Sinopharm/Beijing": "Virus inactivado",
257. "Covaxin": "Virus inactivado",
258. "Valneva": "Virus inactivado",
259. "Novavax": "Proteína recombinante",
260. "Sanofi/GSK": "Proteína recombinante",
261. "SKYCovione": "Proteína recombinante",
262. "Medicago": "Otras"
263. }

264. # Crear nueva columna 'technology' con el tipo de tecnología correspondiente
265. df_vacc2["technology"] = df_vacc2["vaccine"].map(vaccine_technology)

266. # Revisamos que se haya añadido correctamente:
267. print(df_vacc2[["vaccine","technology"]].drop_duplicates().sort_values("vaccine"
     ))
268. # Vemos que en total hay 5 categorías nuevas
269. df_vacc2['technology'].unique()
270. # Vamos a revisar los NaN ahora:
271. df_vacc2.isnull().mean().sort_values(ascending=False)
272. # Perfecto, no hay ningun nan. ya podemos utilizar la database
273. df_vacc2.to_csv("df_vacc2_limpio_para_powerbi.csv", index=False)
```

2 OWID Database vaccinations_*manufacturer.csv:*

```
1.  # Ahora empezamos la limpieza de otra base de datos: vaccinations manufacturer
2.  # Solo hay que poner los mismos paises, controlar las mismas fechas y revisar
    columna de tipos de vacuna
3.  df_vacc = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19
    Project\Databases\vaccinations_manufacturer.csv', encoding = "ISO-8859-1",
    delimiter=',')
4.  df_vacc.shape
5.  df_vacc.info()
6.  df_vacc2 = df_vacc[df_vacc['country'].isin(paises_validos)].copy()
7.  df_vacc2.shape
8.  df_vacc2['country'].unique()
9.  # Esto significa que faltan algunos países en el dataset, simplemente porque no
    habría datos de muchos de ellos
10. # Además el número de filas no ha bajado mucho, por lo que originalmente en el
    dataset había pocos países, muchos ya incluidos en nuestra lsita

11. # Esta es la lista de paises que nos faltan en este dataset y están en el
    primero:
12. # Algeria, australia, brazil, china, colombia, costa rica, egypt, ethiopia,
    fiji, ghana, greece, india, indonesia, iran, israel, kenya, mexico,
13. # Morocco, new zealand, nigeria, panama, papua new guinea, saudi arabia,
    thailand, tunisia, united kingdom, vietnam
```

```python
# Ahora transformamos la fecha y excluimos las que no nos interesan:
# Convertir la columna 'date' a tipo datetime
df_vacc2["date"] = pd.to_datetime(df_vacc2["date"], errors='coerce')

# Filtrar filas solo entre el 1 de marzo de 2020 y el 1 de junio de 2025 (ambos inclusive)
df_vacc2 = df_vacc2[(df_vacc2["date"] >= "2020-03-01") & (df_vacc2["date"] <= "2025-06-01")]

df_vacc2.info()
df_vacc2.shape
df_vacc2['vaccine'].unique() #Vemos como está categorizado la columna vaccine
# Eliminamos los duplicados
df_vacc2.drop_duplicates(inplace=True)
df_vacc2.shape #No hay duplicados
# Me interesa crear una nueva columna para poder asociar cada nombre de vaccine con el tipo de tecnología que usa la propia vacuna
# Diccionario de mapeo vacuna → tecnología
vaccine_technology = {
"Pfizer/BioNTech": "ARNm",
"Moderna": "ARNm",
"Oxford/AstraZeneca": "Vector viral",
"Johnson&Johnson": "Vector viral",
"Sputnik V": "Vector viral",
"Sputnik Light": "Vector viral",
"CanSino": "Vector viral",
"Sinovac": "Virus inactivado",
"Sinopharm/Beijing": "Virus inactivado",
"Covaxin": "Virus inactivado",
"Valneva": "Virus inactivado",
"Novavax": "Proteína recombinante",
"Sanofi/GSK": "Proteína recombinante",
"SKYCovione": "Proteína recombinante",
"Medicago": "Otras"
}

# Crear nueva columna 'technology' con el tipo de tecnología correspondiente
df_vacc2["technology"] = df_vacc2["vaccine"].map(vaccine_technology)

# Revisamos que se haya añadido correctamente:
print(df_vacc2[["vaccine", "technology"]].drop_duplicates().sort_values("vaccine"))
# Vemos que en total hay 5 categorías nuevas
df_vacc2['technology'].unique()
# Vamos a revisar los NaN ahora:
df_vacc2.isnull().mean().sort_values(ascending=False)
# Perfecto, no hay ningun nan. ya podemos utilizar la database
df_vacc2.to_csv("df_vacc2_limpio_para_powerbi.csv", index=False)
```

3 Cleaning of VAERS Database *data.csv*

```
# Author: Miguel Atarés
# coding: utf-8
#   Análisis de la Base de Datos: DATA2021
# </h1>

import pandas as pd

df_VAERS_2021_Data = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19 Project\Databases\2021VAERSData\2021VAERSDATA.csv', encoding = "ISO-8859-1", delimiter=',')

#Nos da un error que indica que algunas columnas son inconsistentes en sus datos (mezclan tipos de datos)
#Para solucionarlo, le vamos a decir que ignore este mensaje y cargue todo normalmente. Ya revisaremos luego los tipos de datos.

df_VAERS_2021_Data = pd.read_csv(
r'C:\Users\migue\Desktop\COVID-19 Project\Databases\2021VAERSData\2021VAERSDATA.csv',
encoding="ISO-8859-1",
delimiter=',',
low_memory=False
)

df_VAERS_2021_Data.shape

df_VAERS_2021_Data.info()

#Lo primero es eliminar columnas que ya analizamos previamente que no nos interesan

# Lista definitiva de columnas que quieres conservar
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"
]

# Filtrar el DataFrame para conservar solo las columnas deseadas
df2_VAERS_2021_Data = df_VAERS_2021_Data[columnas_a_conservar].copy()

df2_VAERS_2021_Data.shape

#Pasamos de (768712, 36) a (768712, 17)

df2_VAERS_2021_Data.drop_duplicates(inplace=True) #Eliminamos duplicados
```

```
df2_VAERS_2021_Data.shape #Había algun duplicado como podemos osbervar

df2_VAERS_2021_Data.info()

df2_VAERS_2021_Data.head()

#Vamos a pasar a datetime las fechas, que son 4 columnas en total:
fechas = ["RECVDATE", "VAX_DATE", "ONSET_DATE", "DATEDIED"]
for col in fechas:
df2_VAERS_2021_Data[col] = pd.to_datetime(df2_VAERS_2021_Data[col],
errors="coerce")

#Las demás columnas están en el formato correcto de datos

categoricas = [
"SEX", "DIED", "L_THREAT", "ER_VISIT", "HOSPITAL",
"DISABLE", "RECOVD", "BIRTH_DEFECT", "STATE"
]

for col in categoricas:
print(f"\n🔍 Valores únicos en la columna '{col}':")
print(df2_VAERS_2021_Data[col].unique())

# Hay inconsistencia en dos columnas. La primera RECOVD con 'U'. Suele ser
'unknown', pero lo vamos a pasar a NaN para que no de problemas
# En STATE hay algunos estados en minustuclas (Ca, Tx, Co) y 4 códigos no
documentados en VAERS (XB, XL, XV, QM)
# El resto de estados de STATE son los 50 americanos, Washington y otros
territorios como puerto rico, guam, islas virgenes, etc

#Primero sustituimos U por NaN:
df2_VAERS_2021_Data["RECOVD"] = df2_VAERS_2021_Data["RECOVD"].replace("U",
pd.NA)

#Ahora solucionamos el problema con la columna STATE:

#1. Corrección de valores sin todo mayusculas
df2_VAERS_2021_Data["STATE"] = df2_VAERS_2021_Data["STATE"].str.upper()

# 2. Conservamos solo los estados válidos
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'
]
```

```
df2_VAERS_2021_Data.loc[~df2_VAERS_2021_Data["STATE"].isin(estados_validos), "STATE"] = pd.NA

#¿Por qué aparece Puerto Rico y otros similares en datasets como VAERS?
#   Puerto Rico (PR), Guam (GU), Islas Vírgenes (VI), Samoa Americana (AS), Islas Marianas del Norte (MP), entre otros,
#   son territorios no incorporados de Estados Unidos. No son estados pero son jurisdicciones bajo soberania estadounidense
#   es decir, son ciudadanos estadounidenses, reciben servicios federales como salud publica y vacunas y participan en programas como VAERS

#Revisamos de nuevo los valores únicos de las columnas categóricas
categoricas = [
"SEX", "DIED", "L_THREAT", "ER_VISIT", "HOSPITAL",
"DISABLE", "RECOVD", "BIRTH_DEFECT", "STATE"
]

for col in categoricas:
print(f"\n🔍 Valores únicos en la columna '{col}':")
print(df2_VAERS_2021_Data[col].unique())

#Eliminamos las filas con NaN en la columna STATE, ya que no tienen sentido
df2_VAERS_2021_Data = df2_VAERS_2021_Data.dropna(subset=["STATE"])

df2_VAERS_2021_Data.shape # Pasamos de (768386, 17) a (651494, 17)

df2_VAERS_2021_Data.isnull().mean().sort_values(ascending=False)

#Veo que hay muchos valores con NaN, pero tiene su explicación, ya que estas columnas significan cosas como:
#   Defectos en nacimiento, fecha de muerte, muerte, amenaza vital, etc. Y si hay un NaN implica que no se ha reportado, no que no se ha producido
#   Es decir, que podemos sustituir algunas de estas columnas los NaN por 'No'

#Vamos a hacer esto con las columnas que claramente si es NaN se puede poner que No:
columnas_binarias = ["DIED", "L_THREAT", "HOSPITAL", "DISABLE", "BIRTH_DEFECT", "ER_VISIT"]

for col in columnas_binarias:
df2_VAERS_2021_Data.loc[:, col] = df2_VAERS_2021_Data[col].fillna("N")

#Perfecto, ahora mucho mejor. Datedied y Hospdays no puedo ponerles que 'No' porque son fechas o días.
df2_VAERS_2021_Data.isnull().mean().sort_values(ascending=False)

#Ahora vamos a revisar cuales son los rangos de fechas que se manejan en el dataset para ver inconsistencias
```

```
#Columnas a analizar:
date_cols = ["RECVDATE", "VAX_DATE", "ONSET_DATE", "DATEDIED"]

# Mostrar rango de fechas por cada columna
for col in date_cols:
print(f"📅 {col} → Desde: {df2_VAERS_2021_Data[col].min()}  hasta: {df2_VAERS_2021_Data[col].max()}")

#Vemos que hay columnas con fechas sin sentido, seguramente de fallos a la hora de pasar datos y tal. Vamos a poner fecha min y max

# Pongo fechas un poco por encima y por abajo por si acaso, aunque la vacunación empezó oficalmente en EEUU el 14 de diciembre con Pfizer. Fecha máxima 2022 porque el dataset es de 2021
fecha_min = pd.to_datetime("2020-12-01")
fecha_max = pd.to_datetime("2022-01-15")
# Vamos a filtrar las fechas para quedarnos solo con las que están dentro de este rango
df2_VAERS_2021_Data = df2_VAERS_2021_Data[
(df2_VAERS_2021_Data["VAX_DATE"].isna() | ((df2_VAERS_2021_Data["VAX_DATE"] >= fecha_min) & (df2_VAERS_2021_Data["VAX_DATE"] <= fecha_max))) &
(df2_VAERS_2021_Data["ONSET_DATE"].isna() | ((df2_VAERS_2021_Data["ONSET_DATE"] >= fecha_min) & (df2_VAERS_2021_Data["ONSET_DATE"] <= fecha_max))) &
(df2_VAERS_2021_Data["DATEDIED"].isna() | ((df2_VAERS_2021_Data["DATEDIED"] >= fecha_min) & (df2_VAERS_2021_Data["DATEDIED"] <= fecha_max)))
]

# Volvemos a revisar las columnas
date_cols = ["RECVDATE", "VAX_DATE", "ONSET_DATE", "DATEDIED"]

# Mostrar rango de fechas por cada columna
for col in date_cols:
print(f"📅 {col} → Desde: {df2_VAERS_2021_Data[col].min()}  hasta: {df2_VAERS_2021_Data[col].max()}")

df2_VAERS_2021_Data.shape # Pasamos de (651494, 17) a (646175, 17), eliminando datos de fechas incorrectos.

df2_VAERS_2021_Data.describe() #He pensado que quizás habría valores numéricos mal
# Todo está bien salvo HOSPDAYS, nadie se pega 99999 dias en el hospital
# Voy a comprobar quien ha estado más de un año en un hospital, las eliminaré y listo

df2_VAERS_2021_Data.shape

# Listo, elimino los que tienen mas de un año porque es o muy raro o falso. Hay que poner los NaN también ya que significan que no han estado en el hospital
df3_VAERS_2021_Data = df2_VAERS_2021_Data[
```

```
(df2_VAERS_2021_Data["HOSPDAYS"] <= 365) |
(df2_VAERS_2021_Data["HOSPDAYS"].isna())
]

df3_VAERS_2021_Data.shape

df3_VAERS_2021_Data.describe()

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset. Repasemos lo hecho o no hecho aquí:
# ✅ 2.1) Limpiar los NaN
# ✅ 2.2) Formato datetime
# ⏳ 2.3) Modificar decimales --> En PBI
# ✅ 2.4) Revisar duplicados
# ✅ 2.5) Revisar columnas categóricas
# ✅ 2.6) Outliers (IQR, boxplots) --> Outliers y medidas extrañas comprobados con describe()
# ❌ 2.7) Distribución de Datos (histogramas, frecuencias) --> Idem, en PBI
# ✅ 2.8) Filtrar fechas previas a la pandemia
# ❌ 2.9) Correlaciones --> PBI
# ✅ 2.10) Eliminar columnas innecesarias
# ✅ 2.11) Revisar valores de desviacion como minimos y maximos de columnas

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset.
df3_VAERS_2021_Data.to_csv("df3VAERS2021Data_limpio_para_powerbi.csv", index=False)
```

Cleaning of VAERS Database *symptoms.csv*

```
# Author: Miguel Atarés
# coding: utf-8
# Análisis de la Base de Datos: SYMPTOMS2021

df_SYMPTOMS_2021_Data = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19 Project\Databases\2021VAERSData\2021VAERSSYMPTOMS.csv', encoding = "ISO-8859-1", delimiter=',')

#Primero eliminamos las columnas:
# Lista de columnas que NO se van a conservar
columnas_a_eliminar = [
"SYMPTOMVERSION1",
"SYMPTOMVERSION2",
"SYMPTOMVERSION3",
"SYMPTOMVERSION4",
"SYMPTOMVERSION5",
"ORDER"
```

```
]

# Eliminar esas columnas del DataFrame
df2_SYMPTOMS_2021_Data = df_SYMPTOMS_2021_Data.drop(columns=columnas_a_eliminar)

df2_SYMPTOMS_2021_Data.shape

df2_SYMPTOMS_2021_Data.head()

#Claro, no hay más variables numéricas realmente
df2_SYMPTOMS_2021_Data.describe()

#Tipo de datos está ok
df2_SYMPTOMS_2021_Data.info()

#limpiamos duplicados
df2_SYMPTOMS_2021_Data.drop_duplicates(inplace=True)

df2_SYMPTOMS_2021_Data.shape #Pasamos de (1030218, 6) a (1028726, 6) por lo que había algún duplicado

#Siempre y cuando esté el ID y el SYMPTOM1 todo está ok.
df2_SYMPTOMS_2021_Data.isnull().mean().sort_values(ascending=False)

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset. Repasemos lo hecho o no hecho aquí:
# ✅ 2.1) Limpiar los NaN --> Los que hay es normal que los haya
# ✅ 2.2) Formato datetime --> No hay
# ✅ 2.3) Modificar decimales --> No hay
# ✅ 2.4) Revisar duplicados
# ✅ 2.5) Revisar columnas categóricas --> No hay
# ✅ 2.6) Outliers (IQR, boxplots) --> No hay
# ❌ 2.7) Distribución de Datos (histogramas, frecuencias) --> PBI
# ✅ 2.8) Filtrar fechas previas a la pandemia --> No hay
# ❌ 2.9) Correlaciones --> PBI
# ✅ 2.10) Eliminar columnas innecesarias
# ✅ 2.11) Revisar valores de desviacion como minimos y maximos de columnas --> No hay

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset.
df2_SYMPTOMS_2021_Data.to_csv("df2VAERS2021Symptoms_limpio_para_powerbi.csv", index=False)
```

5 Cleaning of VAERS Database *VAX.csv*

```
# Author: Miguel Atarés
# coding: utf-8
#   Análisis de la Base de Datos: SYMPTOMS2021

df_SYMPTOMS_2021_Data = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19 Project\Databases\2021VAERSData\2021VAERSSYMPTOMS.csv', encoding = "ISO-8859-1", delimiter=',')

#Primero eliminamos las columnas:
# Lista de columnas que NO se van a conservar
columnas_a_eliminar = [
"SYMPTOMVERSION1",
"SYMPTOMVERSION2",
"SYMPTOMVERSION3",
"SYMPTOMVERSION4",
"SYMPTOMVERSION5",
"ORDER"
]

# Eliminar esas columnas del DataFrame
df2_SYMPTOMS_2021_Data = df_SYMPTOMS_2021_Data.drop(columns=columnas_a_eliminar)

df2_SYMPTOMS_2021_Data.shape

df2_SYMPTOMS_2021_Data.head()

#Claro, no hay más variables numéricas realmente
df2_SYMPTOMS_2021_Data.describe()

#Tipo de datos está ok
df2_SYMPTOMS_2021_Data.info()

#limpiamos duplicados
df2_SYMPTOMS_2021_Data.drop_duplicates(inplace=True)

df2_SYMPTOMS_2021_Data.shape #Pasamos de (1030218, 6) a (1028726, 6) por lo que había algún duplicado

#Siempre y cuando esté el ID y el SYMPTOM1 todo está ok.
df2_SYMPTOMS_2021_Data.isnull().mean().sort_values(ascending=False)

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset. Repasemos lo hecho o no hecho aquí:
# ✅ 2.1) Limpiar los NaN --> Los que hay es normal que los haya
# ✅ 2.2) Formato datetime --> No hay
```

```
# ✅ 2.3) Modificar decimales --> No hay
# ✅ 2.4) Revisar duplicados
# ✅ 2.5) Revisar columnas categóricas --> No hay
# ✅ 2.6) Outliers (IQR, boxplots) --> No hay
# ❌ 2.7) Distribución de Datos (histogramas, frecuencias) --> PBI
# ✅ 2.8) Filtrar fechas previas a la pandemia --> No hay
# ❌ 2.9) Correlaciones --> PBI
# ✅ 2.10) Eliminar columnas innecesarias
# ✅ 2.11) Revisar valores de desviacion como minimos y maximos de columnas --> No hay

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset.
df2_SYMPTOMS_2021_Data.to_csv("df2VAERS2021Symptoms_limpio_para_powerbi.csv", index=False)

#   Análisis de la Base de Datos: VAX2021
# </h1>

df_VAX_2021_Data = pd.read_csv(r'C:\Users\migue\Desktop\COVID-19 Project\Databases\2021VAERSData\2021VAERSVAX.csv', encoding = "ISO-8859-1", delimiter=',')

# Eliminamos columnas que no nos interesan
columnas_a_eliminar_VAX = [
"VAX_LOT",
"VAX_ROUTE",
"VAX_SITE",
"VAX_NAME",
"ORDER"
]

# Eliminar columnas no necesarias
df2_VAX_2021_Data = df_VAX_2021_Data.drop(columns=columnas_a_eliminar_VAX)

df2_VAX_2021_Data.head()

#Quizás en PowerBI las interprete como dosis de refuerzo a partir de la tercera.
df2_VAX_2021_Data['VAX_DOSE_SERIES'].unique()

# Solo necesito las de COVID
df2_VAX_2021_Data['VAX_TYPE'].unique()

# Todo lo que contenga la palabra "COVID19" se elimina de la columna VAX_TYPE
df3_VAX_2021_Data = df2_VAX_2021_Data[df2_VAX_2021_Data["VAX_TYPE"].str.contains("COVID19", case=False, na=False)]
```

```
#Perfecto, ahora tenemmos en nuestro database solo estos valores, pero aun así ahora voy a eliminar la columna porque es redundante
df3_VAX_2021_Data['VAX_TYPE'].unique()

#Elimino columna VAX_TYPE
df4_VAX_2021_Data = df3_VAX_2021_Data.drop(columns=["VAX_TYPE"])

df4_VAX_2021_Data.shape

#Eliminamos duplicados
df4_VAX_2021_Data.drop_duplicates(inplace=True)

df4_VAX_2021_Data.shape #Pasamos de (757765, 3) a (742210, 3)

#Miro tipo de datos. Sinceramente creo que las dosis deberían ser un entero, igual elimino los UNK.
df4_VAX_2021_Data.info()

#La verdad que hay pocos NaN, solo un 0,3%.
df4_VAX_2021_Data.isnull().mean().sort_values(ascending=False)

#Poca categorización aquí de tipos de vacunas, pero bueno, mantenemos.
df4_VAX_2021_Data['VAX_MANU'].unique()

# Vamos a ver el % de 'UNK' en "VAX_DOSE_SERIES" a ver si podemos eliminarlo.
total = len(df4_VAX_2021_Data)

# Filas con 'UNK'
unk_count = (df4_VAX_2021_Data["VAX_DOSE_SERIES"] == "UNK").sum()
unk_pct = round((unk_count / total) * 100, 2)

print(f"'UNK': {unk_count} filas ({unk_pct}%)")

# Tiene un 15,36% que es mucho, por lo que creo que lo voy a dejar en el análisis.
# Si veo que me molesta mucho en power BI a la hora de hacer cuentas, quizás copio y pego la columna y la quito y listo.

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset. Repasemos lo hecho o no hecho aquí:
# ✅ 2.1) Limpiar los NaN --> Los asumimos
# ✅ 2.2) Formato datetime --> No hay
# ✅ 2.3) Modificar decimales --> No hay
# ✅ 2.4) Revisar duplicados
# ✅ 2.5) Revisar columnas categóricas
# ✅ 2.6) Outliers (IQR, boxplots) --> No hay
# ❌ 2.7) Distribución de Datos (histogramas, frecuencias) --> PBI
# ✅ 2.8) Filtrar fechas previas a la pandemia --> No hay
# ❌ 2.9) Correlaciones --> PBI
```

```
# ✅ 2.10) Eliminar columnas innecesarias
# ✅ 2.11) Revisar valores de desviacion como minimos y maximos de columnas --> No hay

# Con esto ya hemos acabado el analisis exploratorio y podemos empezar en Power BI con este dataset.
df4_VAX_2021_Data.to_csv("df4VAERS2021VAX_limpio_para_powerbi.csv", index=False)
```