
- Alumna: Carmen Luna Reyes
- Materia: Introducción a la Ciencia de Datos
- Nombre del profesor: Jaime Alejandro Romero Sierra
- Terea: Limpieza de datos
- Fecha de entrega: 20/10/2025
- Link al repositorio de GitHub:

KarLuna-123/Limpieza-de-la-Base-de-Datos-Spotify-Analysis.

Descripción inicial de la base de datos

Cada fila representa un usuario único de Spotify.

Columnas y sus características:

- user_id → Identificador único de cada

- usuariogender → Género del usuario (Masculino/Femenino/Otro)

- age → Edad del usuario

- country → Ubicación del usuario

- subscription_type → Tipo de suscripción en Spotify (Gratis, Premium, Familiar, Estudiante)

- listening_time → Minutos escuchados por día

- songs_played_per_day → Número de canciones reproducidas diariamente

- skip_rate → Porcentaje de canciones saltadas

- device_type → Dispositivo utilizado (Móvil, Escritorio, Web)

- ads_listened_per_week → Número de anuncios escuchados por semana

- offline_listening → Uso del modo sin conexión

- is_churned → Variable objetivo (0 = Activo, 1 = Perdido)

Explicación de cada etapa.

1.- Carga de la base/dataset

```
#Importar la libreria y cargar la basee de datos
import pandas as pd
df=pd.read_csv('C:/Users/carme/Downloads/df_sucio.csv')
df
```

| user_id | gender | age | country | subscription_type | listening_time | songs_played_per_day | skip_rate | device_type | ads_listened_per_week | offline_listening | is_churned |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.0 | NaN | 54.0 | CA | Free | 26.0 | 23.0 | NaN | Desktop | 31.0 | 0.0 | 1.0 |
| 2.0 | Other | 33.0 | DE | NaN | 141.0 | 62.0 | 0.34 | Web | 0.0 | 1.0 | 0.0 |
| 3.0 | Male | 38.0 | AU | Premium | 199.0 | 38.0 | 0.04 | Mobile | 0.0 | 1.0 | 1.0 |
| 4.0 | Female | NaN | CA | Student | 36.0 | 2.0 | 0.31 | Mobile | 0.0 | 1.0 | 0.0 |
| 5.0 | Other | 29.0 | US | Family | 250.0 | 57.0 | 0.36 | Mobile | 0.0 | 1.0 | 1.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 335.0 | Female | 38.0 | CA | Free | 89.0 | 59.0 | 0.15 | Mobile | 21.0 | 0.0 | 0.0 |
| 878.0 | Female | 45.0 | PK | Free | 221.0 | 6.0 | 0.48 | Mobile | 21.0 | 0.0 | NaN |
| 3255.0 | Male | 35.0 | AU | Free | 263.0 | 16.0 | 0.39 | Mobile | 21.0 | 0.0 | 0.0 |
| 2658.0 | Male | 55.0 | DE | Premium | 14.0 | 80.0 | 0.42 | Mobile | 0.0 | 1.0 | 0.0 |
| 1842.0 | Other | 37.0 | US | Premium | 226.0 | 5.0 | 0.08 | Desktop | 0.0 | NaN | 1.0 |

Se analizan los primeros datos.

```
#VISUALIZAMOS SUS CARACTERISTICAS
df.head()
```

| | user_id | gender | age | country | subscription_type | listening_time | songs_played_per_day | skip_rate | device_type | ads_listened_per_week | offline_listening | is_chu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | NaN | 54.0 | CA | Free | 26.0 | 23.0 | NaN | Desktop | 31.0 | 0.0 | |
| 1 | 2.0 | Other | 33.0 | DE | NaN | 141.0 | 62.0 | 0.34 | Web | 0.0 | 1.0 | |
| 2 | 3.0 | Male | 38.0 | AU | Premium | 199.0 | 38.0 | 0.04 | Mobile | 0.0 | 1.0 | |
| 3 | 4.0 | Female | NaN | CA | Student | 36.0 | 2.0 | 0.31 | Mobile | 0.0 | 1.0 | |
| 4 | 5.0 | Other | 29.0 | US | Family | 250.0 | 57.0 | 0.36 | Mobile | 0.0 | 1.0 | |

```
#VISUALIZAMOS EL NUMERO DE FILLAS Y SUS COLUMNAS
df.shape
```

```
(10171, 12)
```

2.- Revisión inicial de datos faltantes (info, isnull)

Explicación: imprimí info() para tipos y conteo no nulos y guardé la suma de nulos por columna (nulos_por_columna.csv)

```
#VISUALIZAMOS LAS CARATERISSTICAS NUMERICAS Y CATEGORICAS
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10171 entries, 0 to 10170
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   user_id               9866 non-null   float64
 1   gender                9866 non-null   object 
 2   age                   9666 non-null   float64
 3   country               9866 non-null   object 
 4   subscription_type     9866 non-null   object 
 5   listening_time        9668 non-null   float64
 6   songs_played_per_day  9866 non-null   float64
 7   skip_rate             9866 non-null   object 
 8   device_type           9866 non-null   object 
 9   ads_listened_per_week  9667 non-null   float64
 10  offline_listening     9672 non-null   float64
 11  is_churned            9668 non-null   float64
dtypes: float64(7), object(5)
memory usage: 953.7+ KB
```

```
#Se crea una lista de todos los nombres de las columnas
lista_col=df.columns
lista_col
```

```
Index(['user_id', 'gender', 'age', 'country', 'subscription_type',
       'listening_time', 'songs_played_per_day', 'skip_rate', 'device_type',
       'ads_listened_per_week', 'offline_listening', 'is_churned'],
      dtype='object')
```

```
#Vamos a generar un ciclo for que ayude a visualizar que valores únicos vienen en cada columna
lista_col=df.columns
for n in lista_col:
  print(f"la columna {n} tiene de datos:")
  print(df[n].unique())
  print()
```

```
la columna user_id tiene de datos:
[1.000e+00 2.000e+00 3.000e+00 ... 2.087e+03 4.062e+03 5.266e+03]

la columna gender tiene de datos:
[nan 'Other' 'Male' 'Female']

la columna age tiene de datos:
[54. 33. 38. nan 29. 17. 39. 41. 55. 44. 24. 37. 53. 25. 36. 58. 34. 35.
 49. 45. 32. 19. 28. 22. 50. 51. 46. 18. 56. 57. 48. 59. 21. 30. 23. 47.
 52. 43. 27. 31. 42. 16. 40. 20. 26.]

la columna country tiene de datos:
['CA' 'DE' 'AU' 'US' 'UK' nan 'IN' 'FR' 'PK']

la columna subscription_type tiene de datos:
['Free' nan 'Premium' 'Student' 'Family']

la columna listening_time tiene de datos:
[ 26. 141. 199.  36. 250. 219. 289. 210.  50. 278.  86. 113.  24.  45.
 114. 230. 181. 120. 160. 281. 138.  20. 134. 133.  nan 293.  15. 123.
  18. 149.  72. 291. 147.  59. 280.  47. 260. 122. 255.  49.  90. 288.
 238. 241. 124. 175.  41. 222. 121. 277. 107. 198.  95. 262. 169.  25.
  55. 161.  77. 152. 156.  99. 265. 131.  19. 201. 223.  81.  13.  31.
 105. 202. 284. 125. 237. 224. 195. 173.  85. 196.  22. 283.  40. 251.
 240. 215. 119.  17. 174.  84. 272.  21.  35.  32.  11. 194. 217. 271.
...

la columna is_churned tiene de datos:
[ 1.  0. nan]
```

```
#Contar valores nulos en cada columna.
df.isnull().sum()
```

```
user_id                 305
gender                  305
age                     505
country                 305
subscription_type       305
listening_time          503
songs_played_per_day    305
skip_rate               305
device_type             305
ads_listened_per_week   504
offline_listening       499
is_churned              503
dtype: int64
```

```
#Con este comando se crea un booleano que muestra renglones duplicados
df.duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
10166      True
10167     False
10168     False
10169      True
10170     False
Length: 10171, dtype: bool
```

```
#Contar filas duplicadas
df.duplicated().sum()
```

3.- Busqué duplicados por user_id y subcription type ya que no puede haber u usuario con 2 cuantas activas.

-Detectar usuarios con múltiples filas.

```
#Duplicados
df.duplicated(subset=['user_id', 'subscription_type']).sum()
```

```
df2=df.drop_duplicates(subset=['user_id', 'subscription_type'])
df2
```

| | user_id | gender | age | country | subscription_type | listening_time | songs_played_per_day | skip_rate | device_type | ads_listened_per_week | offline_listening | is |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | NaN | 54.0 | CA | Free | 26.0 | 23.0 | NaN | Desktop | 31.0 | 0.0 | |
| 1 | 2.0 | Other | 33.0 | DE | NaN | 141.0 | 62.0 | 0.34 | Web | 0.0 | 1.0 | |
| 2 | 3.0 | Male | 38.0 | AU | Premium | 199.0 | 38.0 | 0.04 | Mobile | 0.0 | 1.0 | |
| 3 | 4.0 | Female | NaN | CA | Student | 36.0 | 2.0 | 0.31 | Mobile | 0.0 | 1.0 | |
| 4 | 5.0 | Other | 29.0 | US | Family | 250.0 | 57.0 | 0.36 | Mobile | 0.0 | 1.0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 10121 | 4062.0 | Female | 49.0 | FR | Family | 195.0 | 39.0 | 0.52 | Web | 0.0 | 1.0 | |
| 10125 | 6500.0 | Female | NaN | PK | Student | 245.0 | 10.0 | 0.5 | Web | 0.0 | 1.0 | |
| 10138 | 5797.0 | Female | 37.0 | DE | NaN | 116.0 | 29.0 | 0.58 | Desktop | 0.0 | 1.0 | |
| 10139 | 5266.0 | Male | 30.0 | US | Premium | 185.0 | 36.0 | 0.15 | Web | 0.0 | 1.0 | |
| 10157 | 6421.0 | Other | 53.0 | US | NaN | 201.0 | 19.0 | 0.15 | Web | 0.0 | 1.0 | |

4.-Sustitucion de los datos NaN

Teniendo en cuenta que no hay que borrar información hay que rellenar los espacio con información no valida, con datos que no afecten el resultado final

```
# Identificar tipos de columnas
num_cols = df2.select_dtypes(include=['number']).columns
cat_cols = df2.select_dtypes(exclude=['number']).columns

# Rellenar valores numéricos con la mediana
for col in num_cols:
  df2[col].fillna(df2[col].median(), inplace=True)

# Rellenar valores categóricos con la moda
for col in cat_cols:
  df2[col].fillna(df2[col].mode()[0], inplace=True)

# Verificar valores nulos restantes
df2.isnull().sum()
```

```
user_id                 0
gender                  0
age                     0
country                 0
subscription_type       0
listening_time          0
songs_played_per_day    0
skip_rate               0
device_type             0
ads_listened_per_week   0
offline_listening       0
is_churned              0
dtype: int64
```

5.- Se cambia todo a minúsculas

```
for col in cat_cols:
  df2[col] = df2[col].str.strip().str.lower()

# Ejemplo: si hay valores similares
for col in cat_cols:
  print(f"\nValores únicos en {col}:")
  print(df2[col].unique())
```

6.- Traducción de textos al español con el uso de rename y los diccionarios

```
#Se crean diccionarios para la traduccion de informacion de las columnas
traduccionG = {
  'gender':'genero',
  'other' : 'otro',
  'male' : 'masculino',
  'female':'femenino'
}
traduccionC = {
  'country' : 'pais',
  'ca' : 'canada',
  'de' : 'alemania',
  'au' : 'australia',
  'us' : 'estados unidos',
  'uk' : 'reino unido',
  'in' : 'india',
  'fr' : 'francia',
  'pk' : 'pakistan'
}
traduccionST = {
  'subscription_type' : 'tipo_de_subscripcion',
  'free' : 'gratis',
  'premium' : 'pago',
  'student' : 'estudiante',
  'family' : 'familiar'
}
traduccionDT = {
  'device_type' : 'tipo_de_dispositivo',
  'desktop' : 'computador',
  'mobile' : 'celular'
}
```

```
#Se traducen los nombre de las columnas usando rename
df2 = df2.rename(columns={'user_id' : 'nombre_de_usuario', 'gender': 'genero', 'age' :
'edad', 'country' : 'pais',
     'subscription_type': 'tipo_de_subscripcion','listening_time': 'tiempo_de_escucha',
'songs_played_per_day': 'canciones_escuchadas_por_dia',
    'skip_rate' : 'omision', 'device_type' : 'tipo_de_dispositivo','ads_listened_per_week'
: 'anuncios_escuchados_por_semana',
    'offline_listening' : 'escucha_sin_conexion', 'is_churned': 'cancelado'
    })
```

```
df2
```

```
#See traduce con los diccionarios
df2['genero'] = df2['genero'].replace(traduccionG)
df2['pais'] = df2['pais'].replace(traduccionC)
df2['tipo_de_subscripcion'] = df2['tipo_de_subscripcion'].replace(traduccionST)
df2['tipo_de_dispositivo'] = df2['tipo_de_dispositivo'].replace(traduccionDT)

df2
```

| | nombre_de_usuario | genero | edad | pais | tipo_de_subscripcion | tiempo_de_escucha | canciones_escuchadas_por_dia | omision | tipo_de_dispositivo | a |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | masculino | 54.0 | canada | gratis | 26.0 | 23.0 | 0.34 | computador | |
| 1 | 2.0 | otro | 33.0 | alemania | pago | 141.0 | 62.0 | 0.34 | web | |
| 2 | 3.0 | masculino | 38.0 | australia | pago | 199.0 | 38.0 | 0.04 | celular | |
| 3 | 4.0 | femenino | 38.0 | canada | estudiante | 36.0 | 2.0 | 0.31 | celular | |
| 4 | 5.0 | otro | 29.0 | estados unidos | familiar | 250.0 | 57.0 | 0.36 | celular | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 10121 | 4062.0 | femenino | 49.0 | francia | familiar | 195.0 | 39.0 | 0.52 | web | |
| 10125 | 6500.0 | femenino | 38.0 | pakistan | estudiante | 245.0 | 10.0 | 0.5 | web | |
| 10138 | 5797.0 | femenino | 37.0 | alemania | pago | 116.0 | 29.0 | 0.58 | computador | |
| 10139 | 5266.0 | masculino | 30.0 | estados unidos | pago | 185.0 | 36.0 | 0.15 | web | |
| 10157 | 6421.0 | otro | 53.0 | estados unidos | pago | 201.0 | 19.0 | 0.15 | web | |

7945 rows × 12 columns

```
#Se hace el cambio de valores
df2['nombre_de_usuario']=df2['nombre_de_usuario'].astype(int)
df2['edad']=df2['edad'].astype(int)
df2['tiempo_de_escucha']=df2['tiempo_de_escucha'].astype(int)
df2['canciones_escuchadas_por_dia']=df2['canciones_escuchadas_por_dia'].astype(
int)
df2['anuncios_escuchados_por_semana']=df2['anuncios_escuchados_por_semana']
.astype(int)
df2['escucha_sin_conexion']=df2['escucha_sin_conexion'].astype(int)
df2['cancelado']=df2['cancelado'].astype(int)
```

7.- SE VERIFICA LA INFORMACION

```
#Se verifica el cambio de valores
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7945 entries, 0 to 10157
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   nombre_de_usuario               7945 non-null   int64
 1   genero                          7945 non-null   object
 2   edad                            7945 non-null   int64
 3   pais                            7945 non-null   object
 4   tipo_de_subscripcion            7945 non-null   object
 5   tiempo_de_escucha               7945 non-null   int64
 6   canciones_escuchadas_por_dia    7945 non-null   int64
 7   omision                         7945 non-null   object
 8   tipo_de_dispositivo             7945 non-null   object
 9   anuncios_escuchados_por_semana  7945 non-null   int64
 10  escucha_sin_conexion            7945 non-null   int64
 11  cancelado                       7945 non-null   int64
dtypes: int64(7), object(5)
memory usage: 806.9+ KB
```

```
#Los datos nulos
df2.isnull().sum()
```

```
nombre_de_usuario                0
genero                           0
edad                             0
pais                             0
tipo_de_subscripcion             0
tiempo_de_escucha                0
canciones_escuchadas_por_dia     0
omision                          0
tipo_de_dispositivo              0
anuncios_escuchados_por_semana   0
escucha_sin_conexion             0
cancelado                        0
dtype: int64
```

Y por ultimo se guarda la base limpia

```
df2.to_csv("Base_limpia_SpotifyDataset.csv", index=False)
```

**Conclusiones del proceso de limpieza de datos**

**Problemas principales encontrados:**

La base presentaba diversos errores comunes en datos reales: valores faltantes en varias columnas, registros duplicados de usuarios, valores atípicos (tiempos de escucha extremadamente altos o negativos), inconsistencias en formatos de país.

**Técnicas aplicadas para solucionarlos:**

Se realizó una revisión general con info() e isnull() para identificar valores nulos, se eliminaron duplicados priorizando las filas más completas, y los valores atípicos se ajustaron.

Se tradujeron valores categóricos al español; se renombraron columnas para mantener consistencia; y se convirtieron tipos de datos (numéricos y fechas) a sus formatos adecuados. Finalmente, se validó que no quedaran duplicados ni incoherencias.

**Aprendizaje obtenido:**

Durante el proceso comprendí la importancia de la limpieza de datos como paso esencial antes del análisis, ya que una base “sucia” puede generar conclusiones erróneas. Aprendí a usar herramientas de *pandas* para detectar y corregir problemas de calidad, a tomar decisiones informadas sobre imputación y normalización, y a documentar cada paso para asegurar transparencia y reproducibilidad en el tratamiento de datos.