**🔍 PASO 1: Exploración inicial de los CSVs (Análisis exploratorio)**

Primero necesitamos conocer el contenido de tus archivos CSV.

**Acciones:**

1. Abre ambos CSVs con un editor de texto plano o Excel para ver:
   * Cuáles son las columnas.

**Mortalidad**:

0 Sexe 89976 non-null string

1 Edat Categórica, rangos 89976 non-null string

2 Data Fecha 89976 non-null string

3 Defuncions observades 89976 non-null Float64

4 Defuncions esperades 89976 non-null Float64

5 Defuncions esperades (Límit inferior) 89976 non-null Int64

6 Defuncions esperades (Límit superior) 89976 non-null Int64

|  |  |  |  |
| --- | --- | --- | --- |
| **Sexo** | str |  |  |
| **Edad** | str | Categórica, rangos |  |
| **Data** | str | Fecha |  |
| **Defunciones\_observadas** | flt |  |  |
| **Defunciones\_esperadas** | flt |  |  |
| **Defunciones\_esperadas\_limite\_sup** | int |  |  |
| **Defunciones\_esperadas\_limite\_inf** | int |  |  |

**Recetas**:

0 any 10 non-null Int64

1 mes 10 non-null Int64

2 codi de la regió sanitària 10 non-null Int64

3 regió sanitària 10 non-null string

4 grup d'edat 10 non-null string

5 sexe 10 non-null string

6 codi del grup ATC nivell 1 10 non-null string

7 grup ATC nivell 1 10 non-null string

8 codi del grup ATC nivell 2 10 non-null string

9 grup ATC nivell 2 10 non-null string

10 codi del grup ATC nivell 3 10 non-null string

11 grup ATC nivell 3 10 non-null string

12 codi del grup ATC nivell 4 10 non-null string

13 grup ATC nivell 4 10 non-null string

14 nombre de receptes 10 non-null Int64

15 nombre d'envasos 10 non-null Int64

16 import íntegre 10 non-null Float64

17 import aportació CatSalut 10 non-null Float64

|  |  |  |  |
| --- | --- | --- | --- |
| **Anho** | Int |  |  |
| **Mes** | Int | 1-12 |  |
| **Código\_reg\_sanitaria** | Int |  |  |
| **Region\_sanitaria** | Str | Id? |  |
| **Grupo\_edad** | Str | Categórica rangos |  |
| **sexo** | Str |  |  |
| **Código\_grupo\_ATC\_nv\_1** | Str |  |  |
| **grupo\_ATC\_nv\_1** | Str |  |  |
| **Código\_grupo\_ATC\_nv\_2** | Str |  |  |
| **grupo\_ATC\_nv\_2** | Str |  |  |
| **Código\_grupo\_ATC\_nv\_3** | Str |  |  |
| **grupo\_ATC\_nv\_3** | Str |  |  |
| **Código\_grupo\_ATC\_nv\_4** | Str |  |  |
| **grupo\_ATC\_nv\_4** | Str |  |  |
| **numero\_receta** | Int |  |  |
| **numero\_envase** | Int |  |  |
| **importe** | Float |  |  |
| **Importe\_aportacion\_cat\_salut** | float |  |  |
|  |  |  |  |

* + Si tienen encabezados claros.

Estaban en catalán, los he traducido al castellano y clarificado (tablas anteriores)

* + Tipos de datos (¿son fechas, números, texto, etc.?).

Tipos de datos ya claros, los he inducido con df.convert\_types() y revisado manualmente. Fechas convertidas a timestamp. Incluida la tabla recetas donde teníamos 2 campos mes y año. Han sido unificados en un único campo fecha aaaa-mm-01.

* + Si hay datos faltantes o inconsistentes.

Valores columnas de rangos de edad les he limpiado el string para que queden solo los rangos (ej 0-2 en lugar de 0-2 anys).

Datos faltantes. Usando df.isna().sum() obtengo la suma de valores nan de cada columna. PROBLEMA con el dataset de recetas no puedo hacerlo, es demasiado grande asique he hecho un bucle donde voy leyendo bloques de 100K filas y hago una suma acumulada de valores nan encontrados.

Para ambos dataset no tengo NULOS.

* + Si hay valores repetidos (posibles claves foráneas o redundancia).

1. Para un análisis más automatizado, puedes usar Python (p. ej., con pandas):

python

CopyEdit

import pandas as pd

df1 = pd.read\_csv('archivo1.csv')

df2 = pd.read\_csv('archivo2.csv')

print(df1.info())

print(df1.describe(include='all'))

print(df2.info())

print(df2.describe(include='all'))

**🧹 PASO 2: Limpieza de datos**

Debes limpiar tus CSVs **antes de cargarlos** en PostgreSQL.

**Cosas a revisar y limpiar:**

* Quitar filas vacías o incompletas.

HECHO

* Uniformar formatos de fechas (YYYY-MM-DD es lo ideal).

HECHO

* Limpiar texto: quitar espacios, mayúsculas/minúsculas consistentes.

HECHO

* Eliminar duplicados si no tienen sentido.

HECHO

* Convertir columnas con "Sí"/"No", "Verdadero"/"Falso", etc., a booleanos.

HECHO

Guarda los CSVs limpios como archivo1\_limpio.csv, etc.

**🧠 PASO 3: Diseño del modelo relacional (Normalización)**

Con base en la inspección anterior, diseña las tablas **respetando las Formas Normales**:

**Principios clave:**

* **Primera Forma Normal (1FN):** No debe haber atributos multivalorados ni repetidos (una celda, un valor).
* **Segunda Forma Normal (2FN):** Todos los campos no clave deben depender de la clave completa (si la clave es compuesta).
* **Tercera Forma Normal (3FN):** No debe haber dependencias transitivas (campos que dependen de campos que no son claves).

**Acciones:**

1. **Identifica las entidades** principales en los CSVs. Ejemplo:
   * Si un CSV tiene columnas como: cliente\_id, cliente\_nombre, cliente\_direccion, pedido\_id, pedido\_fecha, estás mezclando dos entidades: Clientes y Pedidos.
2. **Separa las entidades** en distintas tablas:
   * clientes(cliente\_id PK, nombre, direccion)
   * pedidos(pedido\_id PK, cliente\_id FK, fecha)
3. **Detecta claves primarias y foráneas.**
   * Evita usar valores que pueden cambiar como claves (usa un id numérico autoincremental cuando sea posible).
4. **Diseña el esquema en papel o diagrama** (opcional, pero profesional).

**🗄️ PASO 4: Crear las tablas en PostgreSQL**

Ahora que tienes el modelo listo, crea las tablas.

**Ejemplo:**

sql

CopyEdit

CREATE TABLE clientes (

cliente\_id SERIAL PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

direccion TEXT

);

CREATE TABLE pedidos (

pedido\_id SERIAL PRIMARY KEY,

cliente\_id INTEGER REFERENCES clientes(cliente\_id),

fecha DATE NOT NULL

);

Usa VARCHAR o TEXT según la longitud esperada. Usa DATE, INTEGER, BOOLEAN, etc., en vez de guardar todo como texto.

**📥 PASO 5: Cargar los datos (Load)**

Puedes cargar los CSVs usando COPY o \copy desde psql.

**Ejemplo:**

sql

CopyEdit

-- Desde terminal psql:

\copy clientes(nombre, direccion) FROM 'clientes\_limpio.csv' DELIMITER ',' CSV HEADER;

\copy pedidos(cliente\_id, fecha) FROM 'pedidos\_limpio.csv' DELIMITER ',' CSV HEADER;

⚠️ Asegúrate de que los CSVs estén ordenados para que no haya referencias inválidas.

**✅ PASO 6: Verificación e índices**

Después de cargar los datos, verifica que:

* No haya datos nulos donde no debería.
* Los IDs estén correctamente relacionados.
* Se puede navegar entre tablas (JOINs funcionales).

Agrega índices si necesitas optimizar búsquedas:

sql

CopyEdit

CREATE INDEX idx\_cliente\_id ON pedidos(cliente\_id);

**🎯 EXTRA: Buenas prácticas adicionales**

* Usa **nombres de tabla y columna en minúsculas** y con snake\_case.
* Mantén una convención clara para claves primarias y foráneas.
* Documenta tu modelo (tabla, campo, tipo, descripción).
* Usa CHECK para restricciones lógicas (ej: edad > 0).
* Haz respaldos (pg\_dump) una vez finalizado.

¿Quieres que te ayude a analizar tus CSVs directamente? Si me los compartes, puedo ayudarte a hacer el diseño relacional paso a paso según su contenido.

4o