# 📘 Documentación

**1° demo: asincrónica**

## 1. Tema, problema y solución

Este proyecto simula la gestión de un Tienda a partir de datos sintéticos.  
El objetivo es disponer de un escenario consistente para practicar análisis, visualización y modelado.

## 2. Dataset de referencia: fuente, definición, estructura, tipos y **escala de medición**

**Fuente:** datos generados con fines educativos.  
**Definición:** base que representa un Tienda, con catálogo de productos, registro de clientes y operaciones de venta.

**Productos (productos.csv)** — ~100 filas  
| Campo | Tipo | Escala | |------------------|------|----------| | id\_producto | int | Nominal | | nombre\_producto| str | Nominal | | categoria | str | Nominal | | precio\_unitario| int | Razón |

**Clientes (clientes.csv)** — ~100 filas  
| Campo | Tipo | Escala | |-------------------|------|----------| | id\_cliente | int | Nominal | | nombre\_cliente | str | Nominal | | email | str | Nominal | | ciudad | str | Nominal | | fecha\_alta | date | Intervalo|

**Ventas (ventas.csv)** — ~120 filas  
| Campo | Tipo | Escala | |------------------|------|-----------| | id\_venta | int | Nominal | | fecha | date | Intervalo | | id\_cliente | int | Nominal | | nombre\_cliente | str | Nominal | | email | str | Nominal | | medio\_pago | str | Nominal |

**Detalle\_Ventas (detalle\_ventas.csv)** — ~300 filas  
| Campo | Tipo | Escala | |-------------------|------|---------| | id\_venta | int | Nominal | | id\_producto | int | Nominal | | nombre\_producto | str | Nominal | | cantidad | int | Razón | | precio\_unitario | int | Razón | | importe | int | Razón |

## 3. Información, pasos, pseudocódigo y diagrama del programa (Sprint 1)

En esta etapa, el programa funciona como un **visor interactivo de la documentación**, para que el usuario obtenga rápidamente la información clave del proyecto desde la terminal.

### 3.1 Contenidos accesibles desde el menú

1. **Tema, problema y solución.**
2. **Dataset de referencia.** Resumen de fuente y definición.
3. **Estructura por tabla.** Columnas, tipo y escala de medición.
4. **Escalas de medición.** Descripción y ejemplos.
5. **Sugerencias y mejoras con Copilot.**
6. **Salir.**

### 3.2 Pasos

1. Cargar en memoria los textos de esta documentación (por ejemplo, leyendo este .md o un módulo textos.py).
2. Mostrar un menú numérico con las secciones enumeradas arriba.
3. Según la opción elegida, imprimir el texto correspondiente en pantalla.
4. Permitir volver al menú hasta seleccionar “Salir”.

### 3.3 Pseudocódigo

Inicio Cargar textos/plantillas de documentación en un diccionario Mientras True: Mostrar menú: 1. Tema, problema y solución 2. Dataset de referencia 3. Estructura por tabla (tipo y escala) 4. Escalas de medición 5. Sugerencias y mejoras con Copilot 6. Salir Leer opción Si opción == 1..6: imprimir texto asociado Si opción == 7: romper bucle Fin

### 3.4 Diagrama de flujo: en carpeta

## 4. Sugerencias y mejoras aplicadas con Copilot

* Separar la documentación en **plantillas** reutilizables (por ejemplo, textos.py) y desacoplarla del código del menú.
* Proveer un modo “**búsqueda**” para localizar palabras clave dentro de la documentación (e.g., “precio”, “escala”).
* Agregar una opción “**exportar sección**” para guardar en .txt/.md lo mostrado por pantalla.
* Incluir tests mínimos para el router de opciones (verifica que cada número abra la sección correcta).

**2° demo: asincrónica**

## 5. Estadísticas descriptivas (detalle\_ventas)

Columnas analizadas: cantidad, precio\_unitario, importe.

**Resultados:**

|  | **cantidad** | **precio\_unitario** | **importe** |
| --- | --- | --- | --- |
| count | 343 | 343 | 343 |
| mean | 2.962 | 2654.5 | 7730.08 |
| std | 1.366 | 1308.69 | 5265.54 |
| min | 1 | 272 | 272 |
| 25% | 2 | 1618.5 | 3489 |
| 50% | 3 | 2512 | 6702 |
| 75% | 4 | 3876 | 10231.5 |
| max | 5 | 4982 | 24865 |

**Código utilizado:**

import pandas as pd

detalle = pd.read\_csv('detalle\_ventas.csv')

print(detalle[['cantidad','precio\_unitario','importe']].describe())

## 6. Correlaciones

Columnas analizadas: cantidad, precio\_unitario, importe.

**Resultados:**

|  | **cantidad** | **precio\_unitario** | **importe** |
| --- | --- | --- | --- |
| cantidad | 1 | -0.074 | 0.6 |
| precio\_unitario | -0.074 | 1 | 0.679 |
| importe | 0.6 | 0.679 | 1 |

**Código utilizado:**

import pandas as pd

detalle = pd.read\_csv('detalle\_ventas.csv')

print(detalle[['cantidad','precio\_unitario','importe']].corr())

## 7. Valores extremos

Para este sprint, se observan **valores muy altos o muy bajos** usando mínimos/máximos y listados ordenados.

**Mínimos y máximos:**

| **cantidad\_min\_val** | **cantidad\_max\_val** | **precio\_unitario\_min\_val** | **precio\_unitario\_max\_val** | **importe\_min\_val** | **importe\_max\_val** |
| --- | --- | --- | --- | --- | --- |
| 1 | 5 | 272 | 4982 | 272 | 24865 |

**Top 5 y Bottom 5 de cantidad:**

*Bottom 5*

| **cantidad** |
| --- |
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |

*Top 5*

| **cantidad** |
| --- |
| 5 |
| 5 |
| 5 |
| 5 |
| 5 |

**Top 5 y Bottom 5 de precio\_unitario:**

*Bottom 5*

| **precio\_unitario** |
| --- |
| 272 |
| 272 |
| 272 |
| 272 |
| 469 |

*Top 5*

| **precio\_unitario** |
| --- |
| 4982 |
| 4982 |
| 4973 |
| 4973 |
| 4973 |

**Top 5 y Bottom 5 de importe:**

*Bottom 5*

| **importe** |
| --- |
| 272 |
| 503 |
| 544 |
| 570 |
| 860 |

*Top 5*

| **importe** |
| --- |
| 24865 |
| 23760 |
| 22150 |
| 21430 |
| 21090 |

**Código utilizado:**

import pandas as pd

detalle = pd.read\_csv('detalle\_ventas.csv')

print(detalle[['cantidad','precio\_unitario','importe']].describe()) # min, max, etc.

print(detalle[['cantidad']].sort\_values(by='cantidad', ascending=True).head(5)) # bottom 5

print(detalle[['cantidad']].sort\_values(by='cantidad', ascending=False).head(5)) # top 5

print(detalle[['precio\_unitario']].sort\_values(by='precio\_unitario', ascending=True).head(5))

print(detalle[['precio\_unitario']].sort\_values(by='precio\_unitario', ascending=False).head(5))

print(detalle[['importe']].sort\_values(by='importe', ascending=True).head(5))

print(detalle[['importe']].sort\_values(by='importe', ascending=False).head(5))

## 8. Gráficos

A continuación, se incluyen tres gráficos que permiten visualizar la información de la base de datos del Tienda.

### 8.1 Histograma de cantidades

import pandas as pd

import matplotlib.pyplot as plt

detalle = pd.read\_csv("detalle\_ventas.csv")

# Histograma de cantidades

detalle["cantidad"].plot(kind="hist", bins=5, edgecolor="black", title="Distribución de cantidades")

plt.xlabel("Cantidad")

plt.ylabel("Frecuencia")

plt.show()

### 8.2 Dispersión precio unitario vs importe

import pandas as pd

import matplotlib.pyplot as plt

detalle = pd.read\_csv("detalle\_ventas.csv")

# Dispersión precio unitario vs importe

plt.scatter(detalle["precio\_unitario"], detalle["importe"], alpha=0.6)

plt.title("Relación precio unitario e importe")

plt.xlabel("Precio unitario")

plt.ylabel("Importe")

plt.show()

### 8.3 Boxplot de importes

import pandas as pd

import matplotlib.pyplot as plt

detalle = pd.read\_csv("detalle\_ventas.csv")

# Boxplot de importes

detalle["importe"].plot(kind="box", title="Distribución de importes")

plt.ylabel("Importe")

plt.show()

## 9. Interpretación de resultados

Al analizar los resultados de la **2° demo: asincrónica** (estadísticas descriptivas, correlaciones, valores extremos y gráficos) se observa lo siguiente:

* Las **estadísticas descriptivas** muestran que la cantidad más común de productos por venta está en torno a 2 o 3 unidades, y que el importe promedio de cada línea de venta es de aproximadamente $7.730, con una dispersión considerable
* El análisis de **correlaciones** confirma que el importe depende de la combinación de cantidad y precio unitario: la relación con la cantidad es positiva y moderada, mientras que con el precio unitario es positiva y algo más fuerte. Entre cantidad y precio la correlación es muy baja, lo que indica que son variables independientes
* La revisión de **valores extremos** refleja que existen importes muy altos (más de $20.000), que no son habituales y podrían corresponder a productos de alto precio o a compras de varias unidades en un mismo ticket
* El **histograma de cantidades** confirma que la mayoría de las ventas son de pocas unidades, lo cual coincide con la lógica de un Tienda donde las compras suelen ser chicas y frecuentes
* El **gráfico de dispersión** permite visualizar que, a mayor precio unitario, se obtienen importes más altos, aunque también hay casos de importes elevados combinados con precios unitarios bajos, lo que se explica por mayores cantidades
* El **boxplot de importes** evidencia la concentración de la mayoría de las ventas en valores bajos o medios, y resalta la existencia de outliers (ventas atípicas con importes muy altos)

**En conjunto, estos resultados muestran que el Tienda trabaja principalmente con ventas pequeñas y frecuentes, mientras que los casos extremos de importes altos representan situaciones poco habituales que conviene analizar por separado.**

## 10. Limpieza y preparación de la base de datos

Este sección describe las **operaciones realizadas** para dejar lista la base del Tienda y el **código** utilizado.

Archivos de entrada:

* productos.csv
* clientes.csv
* ventas.csv
* detalle\_ventas.csv

Archivos de salida (limpios):

* productos\_limpio.csv
* clientes\_limpio.csv
* ventas\_limpio.csv
* detalle\_ventas\_limpio.csv

### 10.1 Operaciones aplicadas

**a. Textos y espacios**

* Quitar espacios al inicio/fin en columnas de texto de todas las tablas.
* Estandarizar forma de escritura en columnas de texto clave:
  + categoria (Productos) → Título (ej.: "Golosinas")
  + ciudad (Clientes) → Título (ej.: "Córdoba")

**b. Duplicados y faltantes**

* Eliminar filas con valores faltantes en claves mínimas:
  + Productos: id\_producto
  + Clientes: id\_cliente, nombre\_cliente
  + Ventas: id\_venta, id\_cliente
  + Detalle\_Ventas: id\_venta, id\_producto, cantidad, precio\_unitario
* Quitar duplicados:
  + Productos por id\_producto
  + Clientes por id\_cliente
  + Ventas por id\_venta
  + Detalle\_Ventas por filas idénticas

**c. Fechas**

* Normalizar fecha (Ventas) al formato ISO YYYY-MM-DD (cuando es posible).

**d. Consistencia de importe**

* Asegurar importe = cantidad \* precio\_unitario en Detalle\_Ventas.

**e. Corección de categorías en Productos**

* Unificar variantes/sinónimos (p. ej., “bebida”, “gaseosas” → Bebidas; “aseo” → Limpieza; “dulces/snacks” → Golosinas).
* Si no hay coincidencia en el mapeo, se deja la categoría en formato Título (ej.: “Alimentos”).

**f. Reemplazo de caracteres erróneos en todas las tablas**

* Reemplazar texto mal decodificado por letras con tilde/ñ correctas (p. ej., CÃ³rdoba → Córdoba, PeÃ±a → Peña).
* También se quitan símbolos comunes que aparecen por error (Â, etc.).

### 10.2 Código

import pandas as pd

# Mapeo de categorías

CATEGORIAS\_MAP = {

"bebida": "Bebidas", "bebidas": "Bebidas", "gaseosa": "Bebidas", "gaseosas": "Bebidas",

"limpieza": "Limpieza", "aseo": "Limpieza",

"golosinas": "Golosinas", "dulces": "Golosinas", "snacks": "Golosinas",

"alimentos": "Alimentos"

}

# Reemplazos de caracteres mal decodificados (mojibake)

REEMPLAZOS = {

"Ã¡": "á", "Ã©": "é", "Ãí": "í", "Ã³": "ó", "Ãº": "ú",

"ÃÁ": "Á", "Ã‰": "É", "ÃÍ": "Í", "Ã“": "Ó", "Ãš": "Ú",

"Ã±": "ñ", "Ã‘": "Ñ",

"Ã¼": "ü", "Ãœ": "Ü",

"â€“": "-", "â€”": "-", "â€˜": "'", "â€™": "'",

"â€œ": '"', "â€": '"',

"Â°": "°", "Â·": "·", "Â": ""

}

def limpiar\_objetos(df: pd.DataFrame) -> pd.DataFrame:

"""Quita espacios y corrige caracteres extraños en columnas de texto (object)."""

obj\_cols = df.select\_dtypes(include=["object"]).columns

for c in obj\_cols:

# a) quitar espacios al inicio/fin

df[c] = df[c].astype(str).str.strip()

# b) reemplazos sencillos de mojibake

df[c] = df[c].replace(REEMPLAZOS, regex=True)

return df

# --------------------------

# 2) Carga de archivos

# --------------------------

productos = pd.read\_csv("productos.csv")

clientes = pd.read\_csv("clientes.csv")

ventas = pd.read\_csv("ventas.csv")

detalle = pd.read\_csv("detalle\_ventas.csv")

# --------------------------

# 3) Limpieza básica de texto

# --------------------------

productos = limpiar\_objetos(productos)

clientes = limpiar\_objetos(clientes)

ventas = limpiar\_objetos(ventas)

detalle = limpiar\_objetos(detalle)

# --------------------------

# 4) Duplicados y faltantes

# --------------------------

if "id\_producto" in productos.columns:

productos = productos.dropna(subset=["id\_producto"]).drop\_duplicates(subset=["id\_producto"])

if {"id\_cliente","nombre\_cliente"}.issubset(clientes.columns):

clientes = clientes.dropna(subset=["id\_cliente","nombre\_cliente"]).drop\_duplicates(subset=["id\_cliente"])

if {"id\_venta","id\_cliente"}.issubset(ventas.columns):

ventas = ventas.dropna(subset=["id\_venta","id\_cliente"]).drop\_duplicates(subset=["id\_venta"])

if {"id\_venta","id\_producto","cantidad","precio\_unitario"}.issubset(detalle.columns):

detalle = detalle.dropna(subset=["id\_venta","id\_producto","cantidad","precio\_unitario"]).drop\_duplicates()

# --------------------------

# 5) Fechas en ventas (ISO)

# --------------------------

if "fecha" in ventas.columns:

ventas["fecha"] = pd.to\_datetime(ventas["fecha"], errors="coerce").dt.date.astype("string")

# --------------------------

# 6) Importe consistente

# --------------------------

if {"cantidad","precio\_unitario"}.issubset(detalle.columns):

detalle["cantidad"] = pd.to\_numeric(detalle["cantidad"], errors="coerce").fillna(0)

detalle["precio\_unitario"] = pd.to\_numeric(detalle["precio\_unitario"], errors="coerce").fillna(0)

detalle["importe"] = (detalle["cantidad"] \* detalle["precio\_unitario"]).round(2)

# --------------------------

# 7) Categorías en productos

# --------------------------

if "categoria" in productos.columns:

# pasamos a minúsculas para mapear, y si no matchea, lo dejamos en Título

cat\_lower = productos["categoria"].str.lower()

productos["categoria"] = cat\_lower.map(CATEGORIAS\_MAP).fillna(productos["categoria"].str.title())

# --------------------------

# 8) Guardado final

# --------------------------

productos.to\_csv("productos\_limpio.csv", index=False)

clientes.to\_csv("clientes\_limpio.csv", index=False)

ventas.to\_csv("ventas\_limpio.csv", index=False)

detalle.to\_csv("detalle\_ventas\_limpio.csv", index=False)