Parte 1

Proyecto: Tienda Aurelion

Autor: Leandro Serantes
Tipo de proyecto: Aplicación Interactiva de Análisis de Datos en Streamlit
Lenguaje principal: Python
Archivo principal: aurelion_app.py
Dataset: Clientes, Ventas, Productos y Detalle de Ventas
Repositorio: GitHub – Proyecto Aurelion

🧭 Introducción General

El proyecto **Tienda Aurelion** tiene como propósito desarrollar una aplicación interactiva que permita explorar, analizar y visualizar información comercial de una tienda ficticia mediante técnicas de análisis de datos y visualización con **Python** y **Streamlit**.

La aplicación fue construida de forma modular, integrando distintos componentes de análisis en forma de “Sprints” (módulos temáticos) que abarcan desde la **limpieza y exploración de datos** hasta el **análisis estadístico y correlacional**.
Cada módulo se puede navegar desde la interfaz lateral, ofreciendo una experiencia intuitiva para recorrer el flujo analítico completo.

El dataset principal está conformado por cuatro tablas interrelacionadas:

Archivo	Descripción	Contenido principal
clientes.xlsx	Registro de clientes de la tienda	ID, género, edad, provincia, fecha de alta, etc.
productos.xlsx	Catálogo de productos disponibles	ID de producto, categoría, precio, proveedor
ventas.xlsx	Operaciones de venta realizadas	ID de venta, cliente, fecha, total de compra
detalle_ventas.xlsx	Desglose por ítems de cada venta	ID de venta, producto, cantidad, subtotal

El código fue diseñado para ser **claro, escalable y didáctico**, priorizando la comprensión de los pasos analíticos y la interpretación de los resultados.
Asimismo, se buscó aplicar **buenas prácticas de desarrollo** (indentación uniforme, modularidad, nombres descriptivos, uso de docstrings y tipado con pandas y typing) y garantizar compatibilidad con despliegues en **Streamlit Cloud**.

Parte 2

⚙️ Estructura del Código – aurelion_app.py

El archivo principal de la aplicación, **aurelion_app.py**, concentra toda la lógica del proyecto Tienda Aurelion.
El código fue organizado de forma **modular**, priorizando la legibilidad, la separación conceptual y la escalabilidad de los distintos análisis desarrollados en los Sprints.

🧩 1. Librerías utilizadas

El proyecto emplea un conjunto de librerías de análisis y visualización ampliamente utilizadas en entornos de ciencia de datos:

Librería	Propósito principal
Streamlit	Desarrollar la interfaz interactiva y estructurar la navegación entre secciones.
Pandas	Manipulación y limpieza de los datasets (.xlsx).
NumPy	Operaciones numéricas, cálculos de correlaciones y estadísticas básicas.
Matplotlib / Seaborn	Creación de gráficos de correlación, distribuciones y visualizaciones analíticas.
Pathlib	Manejo de rutas de archivos de forma portable.
Graphviz	Representación gráfica del modelo de datos y relaciones entre tablas.
Textwrap	Formateo y dedentado de textos HTML o Markdown en las secciones informativas.

Todas las librerías fueron declaradas en el archivo requirements.txt para garantizar la correcta instalación en entornos locales y en **Streamlit Cloud**.

🔷 Bloque A – Configuración inicial

Incluye:

- Importación de librerías.
- Configuración del entorno visual de Streamlit (tema, título, logo y estilo general).
- Definición de funciones auxiliares, como check_wrong_types() para identificar columnas con tipos inconsistentes.

🔷 Bloque B – Carga de datos

Se realiza la lectura de los archivos Excel ubicados en la carpeta BD/:

```python
clientes = pd.read_excel("BD/clientes.xlsx")
productos = pd.read_excel("BD/productos.xlsx")
ventas = pd.read_excel("BD/ventas.xlsx")
detalle_ventas = pd.read_excel("BD/detalle_ventas.xlsx")
```

Cada dataset es mostrado en la interfaz mediante tablas y resúmenes estadísticos.
El código también incorpora validaciones y manejo de errores para evitar fallas en caso de datos faltantes o inconsistentes.

🔷 **Bloque C – Navegación interactiva**

Mediante la barra lateral (st.sidebar), el usuario puede seleccionar el **Sprint** o **módulo** que desea visualizar.
Por ejemplo:

```python
section = st.sidebar.radio(
    "Seleccioná una sección",
    ["Inicio", "Sprint 1", "Sprint 2", "Modelo de Datos"]
)
```

Cada opción dispara un bloque de código independiente, con su propio contenido, texto explicativo y visualizaciones.

🔷 **Bloque D – Visualizaciones y análisis**

Se implementan distintas secciones analíticas:

- **Sprint 1:** exploración general, conteo de registros, estructura de datos y detección de inconsistencias.
- **Sprint 2:** análisis de correlaciones numéricas, gráficos de calor con Seaborn, identificación de variables más relacionadas y gráficos de dispersión dinámicos.
- **Modelo de Datos:** visualización de relaciones entre tablas utilizando st.graphviz_chart().

Estos gráficos son completamente interactivos, generados dinámicamente a partir de la selección del usuario.

📦 **3. Buenas prácticas implementadas**

- **Imports unificados y sin duplicados**, optimizando tiempos de carga.

- **Funciones tipadas y documentadas** con pandas.DataFrame para mayor claridad.
- **Indentación y espaciado consistentes** (bloques visualmente equilibrados).
- **Uso de comentarios estructurados** (# ==========) para dividir secciones.
- **Separación lógica de análisis** mediante condicionales if section == ... para facilitar futuras expansiones (como un Sprint 3).
- **Código preparado para despliegue en Streamlit Cloud**, con estructura de carpetas clara (BD/ y IMAGES/).

📄 4. Estructura de carpetas del proyecto

```bash
AURELION/
├── aurelion_app.py
├── requirements.txt
├── AURELION-README.md
├── BD/
│   ├── clientes.xlsx
│   ├── productos.xlsx
│   ├── ventas.xlsx
│   └── detalle_ventas.xlsx
└── IMAGES/
    ├── LOGO.png
    └── LOGO2.png
```

Esta organización asegura la compatibilidad con **Streamlit Cloud**, facilita la navegación del código y mejora la mantenibilidad del proyecto.

Parte 3

🧹 Sprint 1 — Exploración y Limpieza de Datos

El **Sprint 1** tuvo como objetivo realizar una **revisión exploratoria y depuración inicial** de las bases de datos de la Tienda Aurelion, con el fin de asegurar la integridad, coherencia y calidad de los registros antes de avanzar con los análisis posteriores.

🔍 1. Propósito del Sprint

- Identificar y corregir **inconsistencias en tipos de datos** (fechas, números, textos).
- Verificar la **cantidad de registros válidos** por tabla.
- Detectar **valores faltantes o duplicados**.
- Evaluar la **estructura relacional** entre las tablas de clientes, productos y ventas.
- Comprobar la **coherencia de las claves primarias y foráneas**.

Este primer módulo permitió garantizar que todos los archivos .xlsx cargados sean consistentes y compatibles para el análisis cruzado que se desarrolló en el Sprint 2.

🧩 2. Procesos principales implementados

🔶 a) Carga y vista previa de datasets

Cada uno de los archivos fue leído con pandas.read_excel() y mostrado en pantalla con st.dataframe(), permitiendo al usuario navegar los registros y comprobar su estructura.
Se validó la existencia de las columnas clave y se calcularon métricas básicas: número de filas, columnas, tipos de datos y valores nulos.

🔶 b) Validación de tipos de datos

Se creó la función auxiliar check_wrong_types(df) para evaluar la proporción de columnas que podrían convertirse correctamente a formato numérico o de fecha.
Esta función ayudó a detectar columnas **con formato texto que en realidad representaban números o fechas**, lo cual es un error frecuente al trabajar con Excel.

```python
def check_wrong_types(df: pd.DataFrame) -> pd.DataFrame:
    # Analiza columnas tipo 'object' y evalúa su conversión
    # a formato numérico o de fecha
```

🔶 c) Normalización de fechas

Se incorporó una función robusta para parsear fechas en distintos formatos (dd/mm/aa, dd-mm-aaaa, aaaa/mm/dd, etc.), unificando los valores y corrigiendo registros con separadores inconsistentes (., -, /).

🔶 d) Detección de valores nulos y duplicados

Mediante df.isna().sum() y df.duplicated().sum() se identificaron posibles anomalías.
La app muestra alertas visuales cuando existen columnas con altos porcentajes de datos faltantes, ayudando a definir criterios de limpieza (como eliminar, completar o imputar registros).

🔶 e) Integridad relacional

Se comprobó que los **ID de cliente** presentes en ventas.xlsx existan en clientes.xlsx, y que los **ID de producto** referidos en detalle_ventas.xlsx estén en productos.xlsx.
Esto permitió garantizar la coherencia entre las tablas principales y sus relaciones.

📊 3. Resultados y hallazgos

- Se detectaron columnas con formato object que podían transformarse a **fechas válidas** con más del 90 % de precisión.
- Las tablas **clientes** y **productos** no presentaron duplicados, confirmando su rol de **dimensiones limpias**.
- En **ventas** y **detalle_ventas** se hallaron algunos registros con totales o subtotales inconsistentes, lo que fue documentado para su revisión futura.
- El control de tipos permitió establecer una base sólida para los análisis correlacionales del siguiente módulo.

💡 4. Conclusión del Sprint 1

El Sprint 1 logró consolidar una **base de datos confiable y estructurada**, estableciendo un flujo reproducible de validación y limpieza.
El resultado fue un conjunto de datasets listos para el análisis exploratorio, con tipado correcto, relaciones verificadas y sin errores críticos.

Gracias a esta etapa, la aplicación Aurelion pudo avanzar hacia una exploración analítica más profunda, incluyendo correlaciones, patrones de venta y segmentación de clientes.

Parte 4

📈 Sprint 2 — Análisis Avanzado y Correlaciones

El **Sprint 2** tuvo como eje central profundizar el análisis de los datos de la Tienda Aurelion, enfocándose en el **descubrimiento de patrones, relaciones entre variables numéricas y comportamiento de ventas**.
Se incorporaron visualizaciones interactivas, cálculos estadísticos y herramientas gráficas para obtener una comprensión más completa del conjunto de datos.

🧭 1. Objetivos del Sprint

- Evaluar la **correlación entre variables numéricas** de las diferentes tablas (clientes, ventas y detalle_ventas).
- Identificar los **pares de variables con mayor asociación estadística**.
- Implementar **visualizaciones dinámicas** que permitan comparar las relaciones encontradas.
- Desarrollar un **bloque interpretativo** que sintetice los hallazgos más relevantes.

🧩 2. Principales herramientas aplicadas

🔷 Cálculo de correlaciones

Se implementó una rutina que permite seleccionar la **tabla de interés** y el **método de correlación**:

```
python

corr = df[vars_sel].corr(method=metodo)
```

Los métodos disponibles fueron:

- **Pearson:** mide la relación lineal entre variables.
- **Spearman:** evalúa relaciones monótonas, útil ante valores atípicos.
- **Kendall:** alternativa robusta para tamaños de muestra más pequeños.

🔷 Visualización con *heatmap*

Las correlaciones fueron representadas mediante una **matriz de calor** utilizando Seaborn, lo que permitió visualizar de manera inmediata la fuerza y dirección de las relaciones entre variables.

```
python

sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap="vlag",
    annot=True, fmt=".2f"
)
```

Los colores azules representan correlaciones positivas, mientras que los rojizos indican relaciones negativas.
La intensidad del color refleja la magnitud del vínculo.

🔷 Análisis de los pares más correlacionados

Se desarrolló un algoritmo para ordenar las combinaciones de variables según su **correlación absoluta**, mostrando los pares con mayor fuerza estadística.
Esto permitió detectar patrones como, por ejemplo:

- Correlaciones altas entre **cantidad vendida** y **subtotal**.
- Relaciones moderadas entre **edad del cliente** y **monto promedio de compra**.
- Ausencia de correlaciones significativas entre variables demográficas y operativas.

🔷 Gráficos de dispersión (scatter plots)

A partir de los pares más correlacionados, la aplicación ofrece la opción de visualizar un **gráfico de dispersión interactivo** para observar la relación de manera directa.

```python
fig, ax = plt.subplots()
ax.scatter(df[var_x], df[var_y], alpha=0.5)
```

Esto permite verificar visualmente si las correlaciones numéricas coinciden con una tendencia real o si están afectadas por valores extremos.

📊 3. Resultados y hallazgos

- La correlación más alta se observó entre las variables **cantidad** y **subtotal** del detalle de ventas, superando el 0.9.
- Las variables vinculadas a **precio unitario**, **total de venta** y **cantidad vendida** mostraron patrones consistentes entre sí.
- En el segmento de clientes, no se hallaron correlaciones fuertes entre edad, género o antigüedad y el monto total de compra, lo que sugiere **un comportamiento de consumo relativamente homogéneo**.
- Los gráficos de dispersión confirmaron la linealidad de algunas relaciones detectadas por la matriz.

🧠 4. Conclusiones del Sprint 2

El análisis correlacional permitió **identificar las variables clave que determinan el desempeño comercial** de la Tienda Aurelion.
Las ventas se explican principalmente por factores internos del producto (precio y cantidad), mientras que los datos demográficos del cliente tienen un peso menor en el volumen total de compras.

Esta etapa aportó una base sólida para futuros módulos analíticos (Sprint 3), orientados a la **segmentación de clientes, modelos predictivos de compra y visualizaciones de rendimiento**.

Parte 5

📑 Conclusiones Generales del Proyecto Tienda Aurelion

El desarrollo de la aplicación **Tienda Aurelion** representó un proceso integral de análisis de datos, visualización interactiva y aplicación de buenas prácticas de desarrollo en Python.
A lo largo de los distintos Sprints, se logró construir una herramienta funcional, clara y adaptable, capaz de ofrecer una visión profunda del comportamiento comercial de la tienda.

🧭 1. Síntesis del proceso

1. En el **Sprint 1**, se abordó la **limpieza, validación y depuración de los datasets**, asegurando la coherencia entre tablas y la confiabilidad de los datos.
   Se implementaron funciones propias para detección de tipos erróneos y control de valores nulos, lo que estableció una base sólida para los análisis posteriores.
2. En el **Sprint 2**, se realizó un **análisis exploratorio avanzado**, utilizando métricas de correlación y visualizaciones interactivas.
   Esto permitió detectar relaciones significativas entre las variables operativas (precio, cantidad, subtotal, total), revelando las principales fuentes de variabilidad en los ingresos.
3. A lo largo de ambos módulos, se integraron **herramientas visuales y textos explicativos** dentro de la interfaz de Streamlit, favoreciendo la comprensión del usuario y la interpretación de los resultados.

⚙️ 2. Aprendizajes técnicos

Durante el desarrollo del proyecto se aplicaron y consolidaron conocimientos clave en:

- **Manipulación de datos** con pandas y numpy.
- **Visualización de información** con matplotlib y seaborn.
- **Desarrollo de interfaces dinámicas** con streamlit.
- **Validación de calidad de datos**, limpieza y transformación.
- **Gestión de proyectos en GitHub** y despliegue en la nube con **Streamlit Cloud**.
- **Estructuración modular del código**, manteniendo la legibilidad, eficiencia y mantenibilidad.

Estos aprendizajes reflejan la madurez técnica alcanzada en el manejo de herramientas de análisis y la capacidad de integrarlas en una aplicación real y funcional.

📈 3. Resultados obtenidos

- Se logró una **visión unificada del modelo de datos comercial** de la Tienda Aurelion.
- Se comprobó que las variables **precio, cantidad y subtotal** son los principales impulsores del total de ventas.
- Se identificó un comportamiento estable en los patrones de compra, sin grandes diferencias demográficas.
- Se validó la escalabilidad del código, que permite integrar nuevos módulos de análisis (por ejemplo, segmentación o predicción).

💡 4. Mejoras y desarrollos futuros

A partir de la estructura actual, el proyecto puede ampliarse hacia:

- **Sprint 3:** Segmentación de clientes según frecuencia, monto o antigüedad.
- **Sprint 4:** Análisis predictivo de ventas utilizando modelos de machine learning.
- **Sprint 5:** Dashboard integral con KPIs comerciales, mapas interactivos y alertas automáticas.
- Integración con **fuentes de datos externas** (APIs o bases SQL) para simular escenarios reales de negocio.
- Incorporación de **controles de usuario avanzados** (filtros, sliders, selecciones múltiples) para potenciar la interactividad.

🟣 5. Reflexión final

El proyecto **Tienda Aurelion** no solo permitió aplicar conocimientos técnicos en análisis y visualización de datos, sino también consolidar una metodología de trabajo estructurada, escalable y orientada a la calidad del dato.
El resultado final combina rigor analítico, diseño intuitivo y capacidad de comunicación visual, tres pilares esenciales en el ámbito del **Business Intelligence y la Ciencia de Datos**.