* **Nombre del proyecto:** Tienda Aurelion
* **Alumno/a:** [Tu nombre completo]
* **Curso:** [Curso o materia]
* **Docente:** [Nombre del profesor/a]
* **Fecha de entrega:** [Fecha]

### 1.1 Tema, problema y solución

* **Tema:** Inteligencia Artificial para la optimización estratégica de ventas a partir del análisis integrado de datos comerciales en la tienda Aurelion.
* **Problema:** La empresa mantiene un enfoque retrospectivo, sin una visión proactiva del negocio.

La tienda Aurelion recopila datos valiosos sobre clientes, productos y ventas, pero carece de un sistema para analizarlos de forma integrada. Esto conduce a tres problemas centrales:

1. **Desconocimiento del Cliente:** No se identifican patrones de compra, lo que impide reconocer quiénes son los clientes más valiosos o qué productos prefieren.
2. **Estrategias de Venta Genéricas:** Las campañas de marketing y promociones se aplican a todos por igual, resultando en un bajo retorno de inversión y oportunidades de venta perdidas.
3. **Ineficiencia Operativa:** Responder a preguntas de negocio básicas como "¿Cuál fue nuestro producto más vendido el mes pasado?" requiere un esfuerzo manual y consume mucho tiempo.

En esencia, la empresa opera de forma reactiva, mirando el pasado sin tener una hoja de ruta clara para tomar decisiones proactivas que impulsen el crecimiento futuro.

* **Solución:** Creación de un Flujo de Trabajo Analítico para Generar Inteligencia de Negocio

La solución consiste en desarrollar un programa en **Python** que automatice el proceso de análisis de datos, transformando la información cruda en conocimiento accionable. Este proceso, delineado en el pseudocódigo, seguirá estos pasos:

1. **Consolidación de Datos:** Se unificarán las tablas de Clientes, Productos, Ventas y Detalle de Venta en un único conjunto de datos maestro, creando una visión 360° del negocio.
2. **Análisis Descriptivo Automatizado:** El sistema responderá automáticamente a preguntas clave de negocio, tales como:
   * Identificación del top 10 de productos más vendidos.
   * Ranking de los clientes que generan más ingresos.
   * Análisis de ventas por categoría y por ciudad.
   * Visualización de la evolución de las ventas a lo largo del tiempo.
3. **Fundamento para IA:** Este análisis inicial sentará las bases para aplicar técnicas de Inteligencia Artificial (Machine Learning) en una fase posterior, como la segmentación de clientes para personalizar campañas de marketing.

Para la implementación, se utilizará **Python** con librerías como Pandas. **GitHub Copilot** actuará como un asistente de codificación para acelerar el desarrollo, sugerir código para la limpieza, unión de datos y la generación de análisis y visualizaciones.

### 1.2 Dataset de referencia

El conjunto de datos para el Proyecto Aurelion consiste en cuatro tablas interrelacionadas que simulan el entorno transaccional de una empresa minorista.

* **Fuente:** Datos proporcionados por la cátedra del curso, distribuidos en cuatro archivos de Excel (.xlsx).
* **Definición:** El dataset representa un ecosistema de ventas que captura las interacciones entre clientes y productos. Permite analizar el qué (productos), quién (clientes), cuándo (fecha) y cómo (ventas) de cada transacción.
* **Escala:** Es un conjunto de datos de muestra, diseñado para el desarrollo y la prueba. Contiene un volumen controlado de registros:
  + Clientes: 100 registros
  + Productos: 100 registros
  + Ventas: 120 registros
  + Detalle de Venta: 343 registros
* **Estructura del Modelo:** Los datos están organizados siguiendo un **Modelo Estrella** simplificado, con dos tablas de **Dimensiones** (CLIENTES, PRODUCTOS) que describen las entidades, y dos tablas de **Hechos** (VENTAS, DETALLE\_VENTA) que registran los eventos.

**Estructura Detallada de las Tablas**

A continuación, se detalla la estructura y el contenido de cada tabla:

**1. Tabla de Dimensión:**CLIENTES

* Almacena la información demográfica y de registro de cada cliente.

| **Columna** | **Tipo de Dato Esperado** | **Descripción** |
| --- | --- | --- |
| id\_Cli | Entero (Integer) | Identificador único para cada cliente. (Clave Primaria) |
| Nombre | Texto (String) | Nombre y apellido del cliente. |
| Mail | Texto (String) | Dirección de correo electrónico del cliente. |
| Ciudad | Texto (String) | Ciudad de residencia del cliente. |
| FechaDeAlta | Fecha (Datetime) | Fecha en la que el cliente se registró. |

**2. Tabla de Dimensión:**PRODUCTOS

* Funciona como el catálogo de artículos disponibles para la venta.

| **Columna** | **Tipo de Dato Esperado** | **Descripción** |
| --- | --- | --- |
| Id\_Prod | Entero (Integer) | Identificador único para cada producto. (Clave Primaria) |
| Nombre | Texto (String) | Nombre comercial del producto. |
| Categoría | Texto (String) | Categoría a la que pertenece el producto. |
| PrecioUnitario | Decimal (Float) | Precio de venta de una unidad del producto. |

**3. Tabla de Hechos:**VENTAS

* Registra el encabezado de cada transacción, vinculando un cliente a una fecha y un medio de pago.

| **Columna** | **Tipo de Dato Esperado** | **Descripción** |
| --- | --- | --- |
| id\_Vta | Entero (Integer) | Identificador único para cada venta. (Clave Primaria) |
| Fecha | Fecha (Datetime) | Fecha y hora en que se realizó la venta. |
| id\_cliente | Entero (Integer) | Clave foránea que referencia a la tabla CLIENTES. |
| nom\_cliente | Texto (String) | Campo redundante con el nombre del cliente. |
| email | Texto (String) | Campo redundante con el email del cliente. |
| medio\_pago | Texto (String) | Método de pago utilizado (ej. Tarjeta, Efectivo). |

**4. Tabla de Hechos:**DETALLE\_DE\_VENTA

* Contiene las líneas de cada venta, especificando qué productos, en qué cantidad y a qué precio se vendieron.

| **Columna** | **Tipo de Dato Esperado** | **Descripción** |
| --- | --- | --- |
| id\_Vta | Entero (Integer) | Clave foránea que referencia a la tabla VENTAS. |
| id\_Prod | Entero (Integer) | Clave foránea que referencia a la tabla PRODUCTOS. |
| Nom\_Prod | Texto (String) | Campo redundante con el nombre del producto. |
| Cantidad | Entero (Integer) | Número de unidades vendidas del producto. |
| Precio\_Uni | Decimal (Float) | Precio unitario del producto al momento de la venta. |
| Importe | Decimal (Float) | Cálculo total (Cantidad \* Precio\_Uni). |

### 1.3 Información, pasos, pseudocódigo y diagrama del programa

* **Descripción del funcionamiento:**

El programa del proyecto "Aurelion" es una herramienta de análisis de datos desarrollada en Python. Su principal función es consolidar, limpiar y procesar la información de ventas de la tienda Aurelion a partir de los cuatro archivos de datos proporcionados (Clientes, Productos, Ventas y Detalle de Venta). Una vez que los datos están preparados, el programa ejecuta una serie de análisis predefinidos para responder a preguntas clave del negocio, como por ejemplo la identificación de los productos más vendidos o los clientes más valiosos. Finalmente, presenta estos resultados de forma clara en la consola, sentando las bases para una toma de decisiones informada y basada en datos.

* **Pasos del desarrollo:**

El desarrollo del programa siguió una metodología estructurada en cinco pasos clave:

1. **Cargar y Consolidar Datos:**  
   Se desarrolló la lógica inicial para leer los cuatro archivos de Excel utilizando la librería Pandas. En esta fase, se realizaron las tareas de limpieza (manejo de nulos, eliminación de duplicados) y la unión (merge) de las tablas para crear un DataFrame maestro unificado, que contiene toda la información necesaria para el análisis.
2. **Crear Funciones de Búsqueda y Análisis:**  
   El núcleo del análisis se encapsuló en funciones modulares y reutilizables. Se creó una función específica para cada pregunta de negocio (ej. obtener\_top\_productos(), calcular\_ventas\_por\_categoria()). Este enfoque permite que el código sea más limpio, fácil de mantener y probar.
3. **Interfaz Interactiva en Consola:**  
   Para que el usuario pueda acceder a los análisis, se implementó un menú simple basado en texto en la consola. El programa presenta una lista de opciones numeradas, y el usuario puede seleccionar qué análisis desea ejecutar.
4. **Validación de Errores:**  
   Se añadió una capa de robustez al programa mediante la gestión de errores. Se implementaron bloques try-except para manejar problemas comunes, como no encontrar los archivos de datos en la ruta especificada o que el usuario ingrese una opción no válida en el menú.
5. **Integración con Copilot:**  
   A lo largo de todo el desarrollo, se utilizó GitHub Copilot como un asistente de programación. Su rol fue fundamental para acelerar la escritura de código repetitivo (boilerplate), sugerir sintaxis para operaciones complejas en Pandas y generar rápidamente la estructura base de las funciones y del menú interactivo.

* **Pseudocódigo:**

El siguiente pseudocódigo representa la estructura modular y de alto nivel del programa. El proceso principal invoca a una serie de subprocesos, donde cada uno es responsable de una etapa específica del flujo de análisis.

INICIO\_PROGRAMA

// =================================================================

// PASO 1: CONFIGURACIÓN E IMPORTACIÓN DE LIBRERÍAS

// =================================================================

// En Python, esto sería: import pandas as pd, matplotlib.pyplot as plt, etc.

IMPORTAR librería para manipulación de datos (ej. Pandas)

IMPORTAR librería para visualización (ej. Matplotlib, Seaborn)

// =================================================================

// PASO 2: CARGA DE DATOS

// =================================================================

// Cargar cada tabla desde su fuente (ej. un archivo CSV, Excel o base de datos)

// en una estructura de datos tipo tabla (DataFrame en Pandas).

DEFINIR df\_clientes = CARGAR\_DATOS("ruta/a/clientes.csv")

DEFINIR df\_productos = CARGAR\_DATOS("ruta/a/productos.csv")

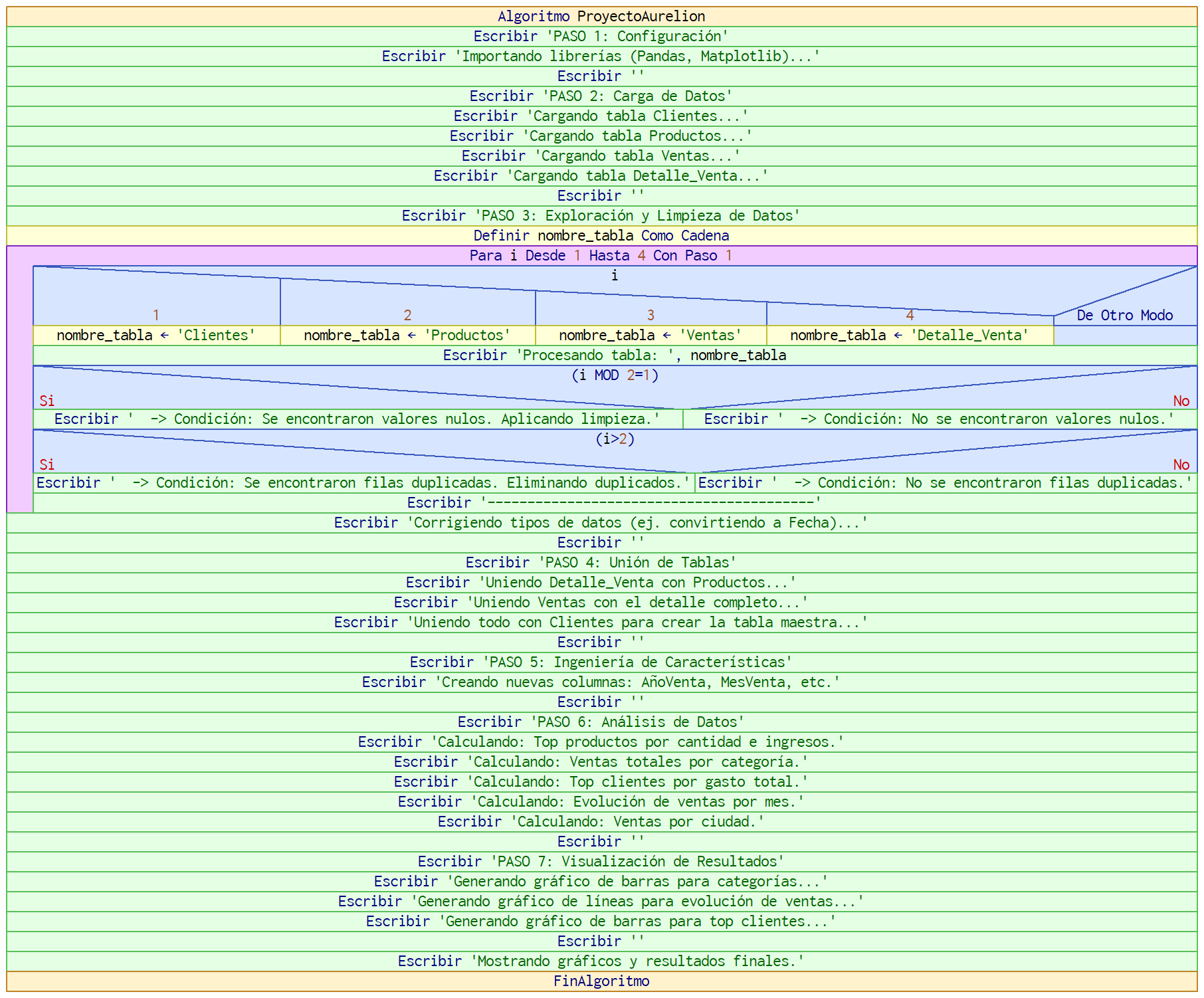
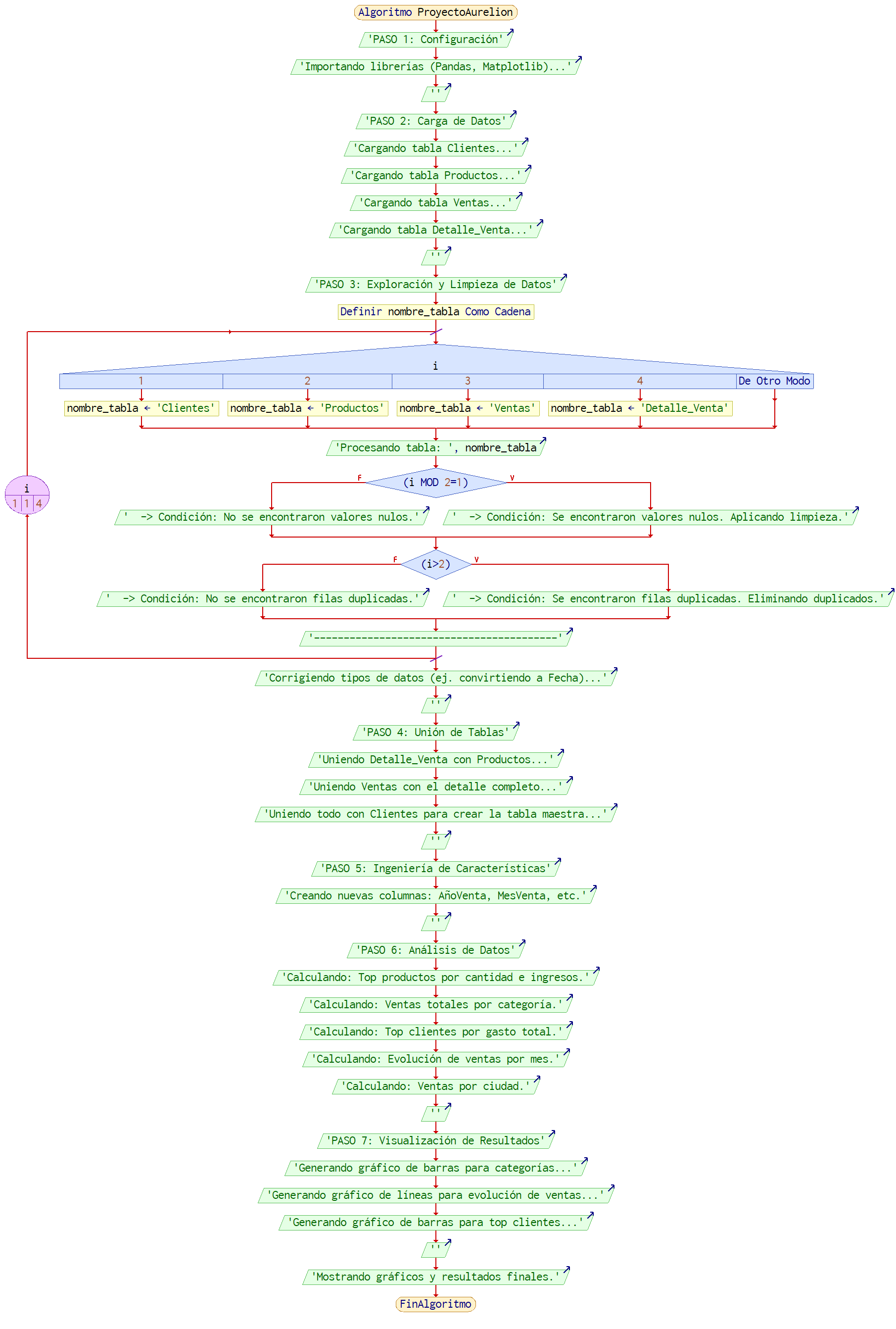
DEFINIR df\_ventas = CARGAR\_DATOS("ruta/a/ventas.csv")

DEFINIR df\_detalle\_venta = CARGAR\_DATOS("ruta/a/detalle\_venta.csv")

**(Se anexa pseudocódigo completo)**

* **Diagrama de flujo:**

El siguiente diagrama de flujo, generado en PSeInt, ilustra la secuencia lógica del pseudocódigo modular. Representa el flujo principal del programa, donde cada bloque rectangular con un borde doble simboliza la ejecución de un subproceso que contiene las operaciones detalladas de esa etapa.



### 1.4 Sugerencias y mejoras aplicadas con Copilot

Durante el desarrollo del script en Python, GitHub Copilot actuó como un asistente proactivo. Sin embargo, su uso requirió un juicio crítico para asegurar que el código final fuera eficiente, legible y se ajustara a los objetivos del proyecto.

* **Sugerencias Aceptadas:**
  + **Autocompletado de Código en Pandas:** Al escribir operaciones de agregación, Copilot demostró ser extremadamente útil. Por ejemplo, después de escribir df\_master.groupby('Categoría'), sugirió correctamente la continuación de la cadena de métodos: ['Importe'].sum().sort\_values(ascending=False). Aceptar esta sugerencia aceleró significativamente el desarrollo del paso de análisis.
  + **Generación de Plantillas para Visualización:** Para la creación de gráficos con Matplotlib y Seaborn, Copilot generó la estructura base (el "boilerplate") de cada gráfico. Tras escribir un comentario como # Gráfico de barras para ventas por ciudad, sugirió el bloque completo sns.barplot(...), plt.title(...), plt.xlabel(...), plt.ylabel(...), lo que permitió centrarse únicamente en la personalización de los detalles estéticos.
* **Sugerencias Descartadas:**
  + **Uso de Funciones Excesivamente Complejas:** En algunas ocasiones, para tareas simples de filtrado, Copilot sugirió el uso de funciones lambda anidadas o list comprehensions complejas. Se decidió descartar estas sugerencias en favor de métodos de Pandas más explícitos y legibles (como el encadenamiento de .loc[] o .groupby()), priorizando la claridad y el fácil mantenimiento del código para un proyecto de este alcance.
  + **Sugerencias Fuera de Alcance:** Al principio del paso de análisis, Copilot sugirió implementar un modelo de machine learning (como K-Means para segmentar clientes) directamente en el script. Aunque es una excelente idea para una fase futura, se descartó porque el objetivo principal de esta etapa inicial es realizar un **análisis descriptivo** por ahora.

## ****Programa en Python****

* Archivo: proyecto\_aurelion.py
* Funcionalidad: El programa es un script de consola que ejecuta el flujo completo de análisis de datos. Primero, carga, limpia y unifica los datos de las cuatro fuentes en una única tabla maestra. Luego, en lugar de simplemente terminar, presenta al usuario un menú interactivo para que pueda consultar los resultados de los análisis clave sin necesidad de ejecutar todo el código de nuevo. Permite visualizar los productos más vendidos, las ventas por categoría, los clientes más importantes y la distribución de las ventas.
* Interacción: La interacción con el programa se realiza dentro de un entorno de **Jupyter Notebook (**.ipynb**)**, lo que facilita un análisis exploratorio y documentado. El flujo de trabajo no se basa en un menú, sino en la **ejecución secuencial de celdas** de código y texto.
* El notebook está estructurado como un informe narrativo donde cada paso del proceso (carga, limpieza, análisis, visualización) está encapsulado en su propia celda o conjunto de celdas. La interacción del usuario consiste en:
* **Leer la Documentación (Markdown):** Cada bloque de código está precedido por una celda de texto (Markdown) que explica el propósito y la lógica de la operación que se va to realizar.
* **Ejecutar Celdas de Código:** El usuario ejecuta cada celda de código de forma individual y en orden (Shift + Enter).
* **Observar la Salida Inmediata (Inline):** El resultado de cada operación —ya sea la previsualización de una tabla (.head()), un resumen informativo (.info()) o un gráfico completo— se muestra inmediatamente debajo de la celda que lo generó.
* Este enfoque **paso a paso** permite al usuario controlar el ritmo del análisis, inspeccionar el estado de los datos en cada etapa y modificar fácilmente cualquier parámetro para experimentar en tiempo real, lo cual es ideal para el desarrollo y la validación de un flujo de análisis de datos.

Anexos:

Pseudocódigo:

INICIO\_PROGRAMA

// =================================================================

// PASO 1: CONFIGURACIÓN E IMPORTACIÓN DE LIBRERÍAS

// =================================================================

// En Python, esto sería: import pandas as pd, matplotlib.pyplot as plt, etc.

IMPORTAR librería para manipulación de datos (ej. Pandas)

IMPORTAR librería para visualización (ej. Matplotlib, Seaborn)

// =================================================================

// PASO 2: CARGA DE DATOS

// =================================================================

// Cargar cada tabla desde su fuente (ej. un archivo CSV, Excel o base de datos)

// en una estructura de datos tipo tabla (DataFrame en Pandas).

DEFINIR df\_clientes = CARGAR\_DATOS("ruta/a/clientes.csv")

DEFINIR df\_productos = CARGAR\_DATOS("ruta/a/productos.csv")

DEFINIR df\_ventas = CARGAR\_DATOS("ruta/a/ventas.csv")

DEFINIR df\_detalle\_venta = CARGAR\_DATOS("ruta/a/detalle\_venta.csv")

// =================================================================

// PASO 3: EXPLORACIÓN Y LIMPIEZA DE DATOS (EDA - Exploratory Data Analysis)

// =================================================================

// Antes de analizar, es crucial entender y limpiar los datos.

PARA CADA tabla EN [df\_clientes, df\_productos, df\_ventas, df\_detalle\_venta]:

MOSTRAR "Información de la tabla:" + nombre\_de\_la\_tabla

MOSTRAR PRIMERAS\_FILAS(tabla) // Ver cómo lucen los datos

MOSTRAR INFO\_GENERAL(tabla) // Resumen de columnas, tipos de datos y nulos

// Verificación de valores nulos

SI HAY\_NULOS(tabla):

MOSTRAR "Se encontraron valores nulos en " + nombre\_de\_la\_tabla

// Decidir una estrategia: eliminar filas, rellenar valores, etc.

// EJEMPLO: RELLENAR\_NULOS(tabla['columna\_con\_nulos'], con\_valor='Desconocido')

FIN SI

// Verificación de duplicados

SI HAY\_DUPLICADOS(tabla):

MOSTRAR "Se encontraron filas duplicadas en " + nombre\_de\_la\_tabla

ELIMINAR\_DUPLICADOS(tabla)

FIN SI

FIN PARA

// Limpieza específica por tabla

// Corregir tipos de datos

CONVERTIR\_A\_FECHA(df\_clientes['FechaDeAlta'])

CONVERTIR\_A\_FECHA(df\_ventas['Fecha'])

// Observación: La tabla VENTAS tiene 'nom\_cliente' y 'email' que son redundantes

// si ya tenemos la tabla CLIENTES. Es una buena práctica usar solo el id\_cliente

// para unir los datos y evitar inconsistencias.

// =================================================================

// PASO 4: UNIÓN DE TABLAS (MERGE / JOIN)

// =================================================================

// Para un análisis completo, necesitamos combinar las tablas en una sola tabla maestra.

// Unir detalle de venta con productos para tener la categoría y más detalles del producto.

DEFINIR df\_detalle\_completo = UNIR(df\_detalle\_venta, df\_productos, en='id\_Prod')

// Unir las ventas con el detalle completo de cada venta.

// La tabla df\_ventas es la "cabeza" de la venta, y df\_detalle\_completo son las "líneas" de esa venta.

DEFINIR df\_ventas\_detalle = UNIR(df\_ventas, df\_detalle\_completo, en='id\_Vta')

// Finalmente, unir la información de los clientes a cada venta.

DEFINIR df\_master = UNIR(df\_ventas\_detalle, df\_clientes, en='id\_cliente')

// Ahora 'df\_master' contiene toda la información en una sola tabla, lista para el análisis.

// Columnas en df\_master: id\_Vta, Fecha, id\_cliente, medio\_pago, id\_Prod, Cantidad, Importe,

// Categoría, Ciudad, FechaDeAlta, etc.

MOSTRAR "Vista previa de la tabla maestra unificada:"

MOSTRAR PRIMERAS\_FILAS(df\_master)

// =================================================================

// PASO 5: INGENIERÍA DE CARACTERÍSTICAS (FEATURE ENGINEERING)

// =================================================================

// Crear nuevas columnas a partir de las existentes para facilitar el análisis.

EXTRAER AÑO de df\_master['Fecha'] y guardarlo en una nueva columna 'AñoVenta'

EXTRAER MES de df\_master['Fecha'] y guardarlo en una nueva columna 'MesVenta'

EXTRAER DÍA\_DE\_SEMANA de df\_master['Fecha'] y guardarlo en una nueva columna 'DiaSemanaVenta'

// =================================================================

// PASO 6: ANÁLISIS DE DATOS (RESPONDER PREGUNTAS DE NEGOCIO)

// =================================================================

// Aquí es donde extraemos insights valiosos.

// Pregunta 1: ¿Cuáles son los productos más vendidos (en cantidad)?

DEFINIR top\_productos\_cantidad = df\_master.AGRUPAR\_POR('Nom\_Prod').SUMAR('Cantidad')

DEFINIR top\_productos\_cantidad\_ordenado = ORDENAR(top\_productos\_cantidad, por='Cantidad', descendente=VERDADERO)

MOSTRAR "Top 10 productos más vendidos por cantidad:", top\_productos\_cantidad\_ordenado.PRIMEROS(10)

// Pregunta 2: ¿Cuáles son los productos que generan más ingresos (en importe)?

DEFINIR top\_productos\_ingresos = df\_master.AGRUPAR\_POR('Nom\_Prod').SUMAR('Importe')

DEFINIR top\_productos\_ingresos\_ordenado = ORDENAR(top\_productos\_ingresos, por='Importe', descendente=VERDADERO)

MOSTRAR "Top 10 productos con mayores ingresos:", top\_productos\_ingresos\_ordenado.PRIMEROS(10)

// Pregunta 3: ¿Cuáles son las categorías de productos más populares?

DEFINIR ventas\_por\_categoria = df\_master.AGRUPAR\_POR('Categoría').SUMAR('Importe')

DEFINIR ventas\_por\_categoria\_ordenado = ORDENAR(ventas\_por\_categoria, por='Importe', descendente=VERDADERO)

MOSTRAR "Ventas totales por categoría:", ventas\_por\_categoria\_ordenado

// Pregunta 4: ¿Qué clientes compran más (top clientes por importe total)?

DEFINIR top\_clientes = df\_master.AGRUPAR\_POR(['id\_cliente', 'Nombre']).SUMAR('Importe')

DEFINIR top\_clientes\_ordenado = ORDENAR(top\_clientes, por='Importe', descendente=VERDADERO)

MOSTRAR "Top 10 clientes por valor de compra:", top\_clientes\_ordenado.PRIMEROS(10)

// Pregunta 5: ¿Cómo han evolucionado las ventas a lo largo del tiempo (por mes)?

DEFINIR ventas\_por\_mes = df\_master.AGRUPAR\_POR(['AñoVenta', 'MesVenta']).SUMAR('Importe')

MOSTRAR "Evolución de ventas por mes:", ventas\_por\_mes

// Pregunta 6: ¿Desde qué ciudades se realizan más compras?

DEFINIR ventas\_por\_ciudad = df\_master.AGRUPAR\_POR('Ciudad').SUMAR('Importe')

DEFINIR ventas\_por\_ciudad\_ordenado = ORDENAR(ventas\_por\_ciudad, por='Importe', descendente=VERDADERO)

MOSTRAR "Top ciudades por volumen de ventas:", ventas\_por\_ciudad\_ordenado.PRIMEROS(5)

// =================================================================

// PASO 7: VISUALIZACIÓN DE RESULTADOS

// =================================================================

// Un gráfico comunica los resultados de forma más efectiva.

CREAR\_GRAFICO\_DE\_BARRAS(datos=ventas\_por\_categoria\_ordenado, titulo="Ventas por Categoría")

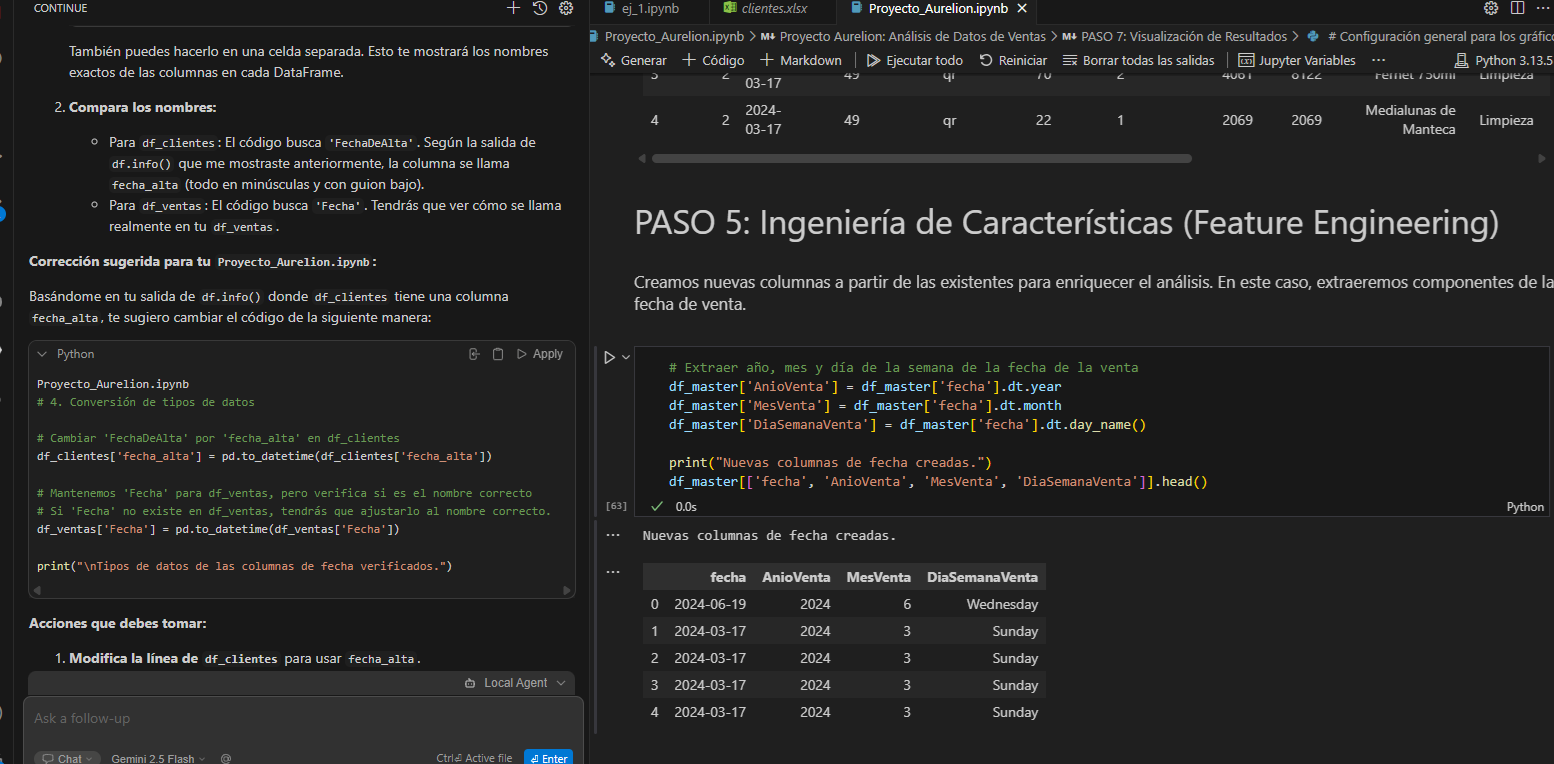
CREAR\_GRAFICO\_DE\_LINEAS(datos=ventas\_por\_mes, titulo="Evolución de Ventas Mensuales")

CREAR\_GRAFICO\_DE\_BARRAS\_HORIZONTALES(datos=top\_clientes\_ordenado.PRIMEROS(10), titulo="Top 10 Clientes")

CREAR\_GRAFICO\_DE\_TARTA(datos=ventas\_por\_ciudad\_ordenado.PRIMEROS(5), titulo="Distribución de Ventas por Ciudad")

MOSTRAR\_GRAFICOS()

FIN\_PROGRAMA



.