Informe de Proyecto:

Módulo 3: Obtención y Preparación de Datos

Nombre: Camila Garrido

1. Introducción

El presente proyecto responde a la necesidad del Equipo de Analítica de Datos de una empresa de e-commerce para resolver la falta de estructura y calidad en su información. Los datos originales presentaban valores nulos, duplicados y outliers.

Objetivo: El objetivo principal del proyecto es desarrollar un proceso automatizado y eficiente para la obtención, limpieza, transformación, análisis y estructuración de datos utilizando Python y las librerías NumPy y Pandas. Al finalizar el proyecto, se espera contar con un dataset limpio, confiable y estructurado, listo para ser utilizado en procesos de análisis y toma de decisiones en la organización.

2. Justificación de las Herramientas Utilizadas

2.1 ¿Por qué NumPy? (leccion_1_numpy.py)

NumPy fue la elección técnica para la generación de datos sintéticos y el cálculo de métricas iniciales debido a:

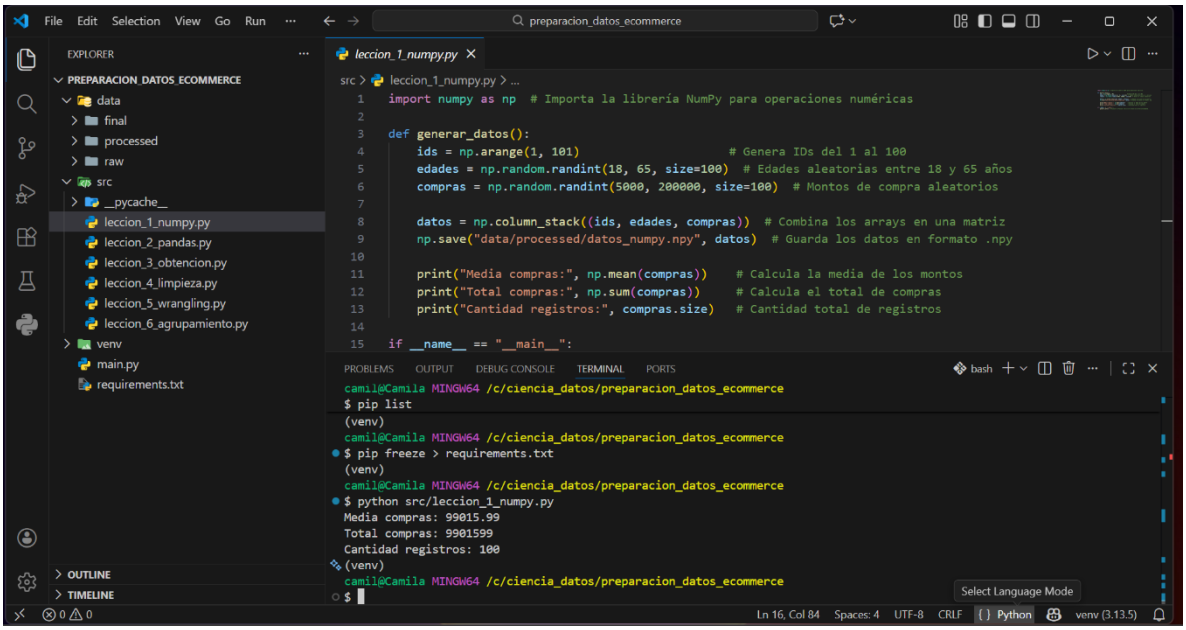

Imagen 1. Muestra la lección_1_numpy.py

- NumPy es eficiente porque guarda los datos ordenados y juntos en la memoria de la computadora, como una fila perfecta. Esto permite que la computadora procese todos los números de una sola vez (como una máquina rápida) en lugar de revisar uno por uno, lo que ahorra mucho tiempo cuando manejamos miles de datos
- Rendimiento: El manejo de arrays es significativamente más rápido que las listas nativas de Python.
- Funciones Vectorizadas: Permite realizar cálculos de media (np.mean), suma (np.sum) y conteo de forma masiva sin usar bucles for, reduciendo el riesgo de errores y optimizando la memoria.

2.2 ¿Por qué Pandas? (leccion_2_pandas.py en adelante)

Pandas es la herramienta principal para la estructuración y limpieza por su versatilidad:

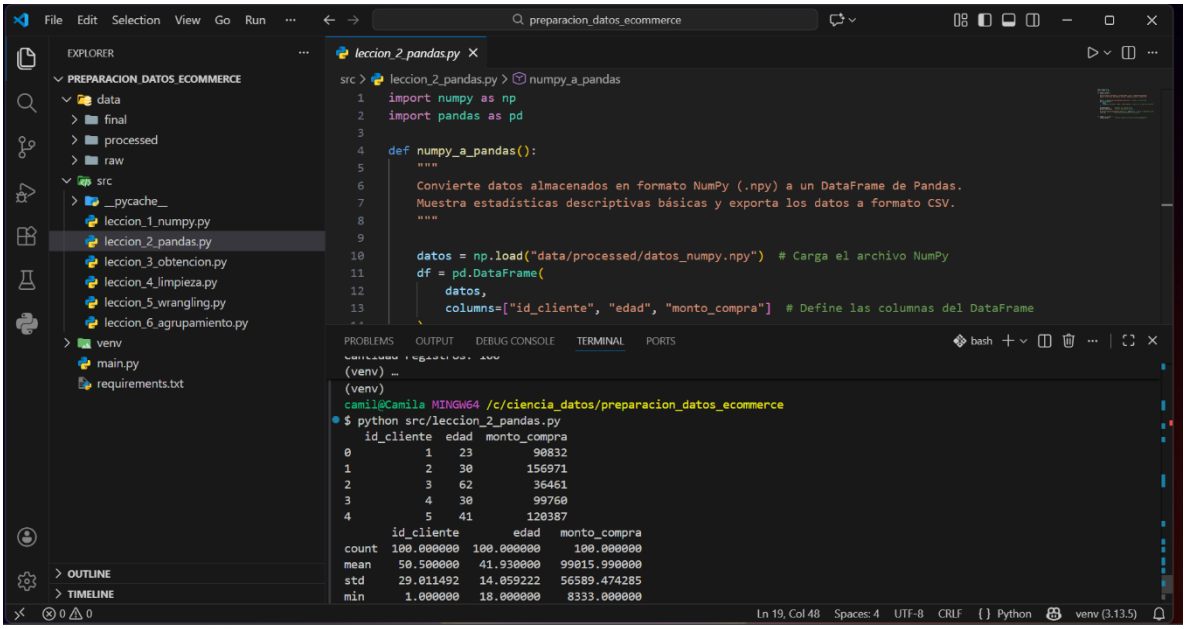

Imagen 2: Muestra la lección_2_pandas.py

- Estructura de DataFrame: Facilita la manipulación de datos tabulares con etiquetas claras. Su estructura de DataFrame permite manejar datos heterogéneos (números, fechas y texto) de forma simultánea, algo que los arrays de NumPy no permiten fácilmente
- Conectividad: Capacidad nativa para leer múltiples formatos (CSV, Excel, HTML) en pocas líneas de código.
- Flexibilidad: Herramientas potentes para el manejo de valores faltantes y transformaciones complejas (Wrangling).

3. Descripción del Dataset y Fuentes de Datos

El proyecto integra datos de diversa naturaleza para simular un entorno real de e-commerce:

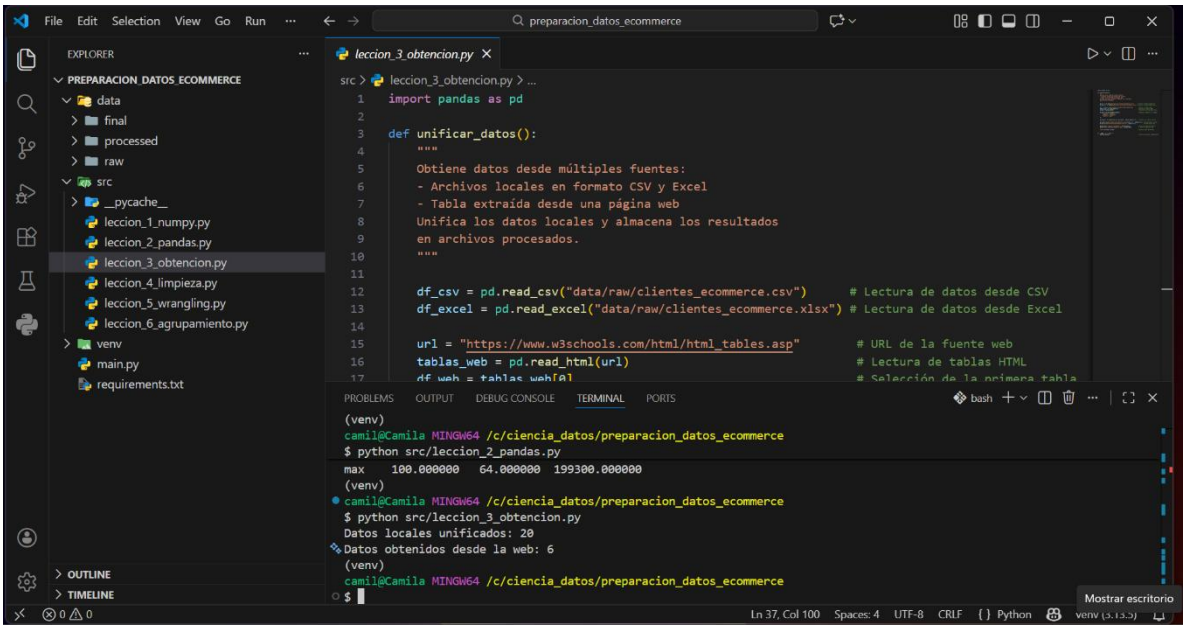

Imagen 3: Muestra la lección_3_obtencion.py

1. Datos Sintéticos: Generados con NumPy (ID, edad, monto de compra), proporcionando una base controlada para pruebas.
2. Fuentes Locales: Archivos clientes_ecommerce.csv y clientes_ecommerce.xlsx que contienen el histórico de la empresa.
3. Fuentes Externas (Web): Extracción de tablas desde la web mediante pd.read_html, permitiendo enriquecer el perfil de los clientes con datos de contacto y país.
4. Desafío encontrado: Los datos web a menudo requieren limpieza de caracteres especiales o renombrado de columnas, a diferencia de los archivos locales que suelen ser más predecibles

4. Técnicas Aplicadas para la Limpieza y Transformación

4.1 Limpieza de Datos (leccion_4_limpieza.py)

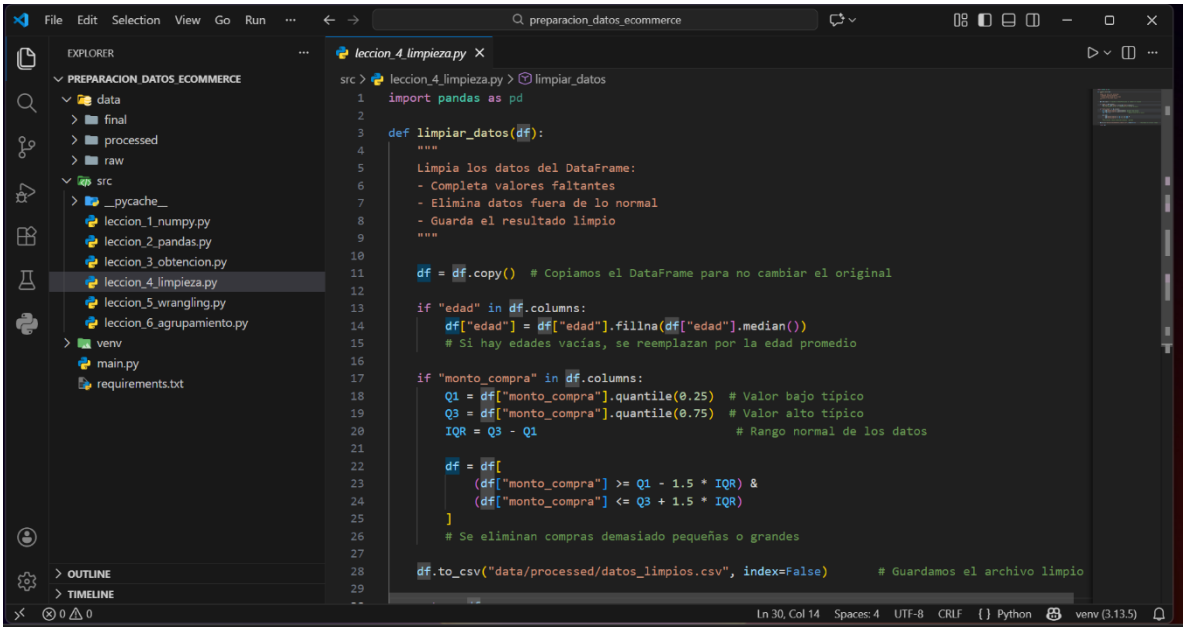

Imagen 4: Muestra la lección_4_limpieza.py

- Tratamiento de Nulos: Se aplicó la técnica de imputación por mediana en la columna edad. Se eligió la mediana sobre el promedio porque es más robusta frente a valores extremos.
- Detección de Outliers: Se utilizó el Rango Intercuartílico (IQR):

  Justificación: Eliminar valores por encima de $Q3 + 1.5 \times IQR$ o por debajo de $Q1 - 1.5 \times IQR$ asegura que los promedios de compra no se vean inflados por errores de carga o transacciones atípicas.

  Robustez Estadística: A diferencia del Z-score, el IQR no asume que los datos siguen una distribución normal, lo cual es ideal para el e-commerce donde los montos de compra suelen ser asimétricos.

  Resistencia a Extremos: El IQR utiliza cuartiles (Q1 y Q3), lo que permite identificar valores atípicos sin que la media o la desviación estándar se vean distorsionadas por los propios errores de carga.

4.2 Data Wrangling (leccion_5_wrangling.py)

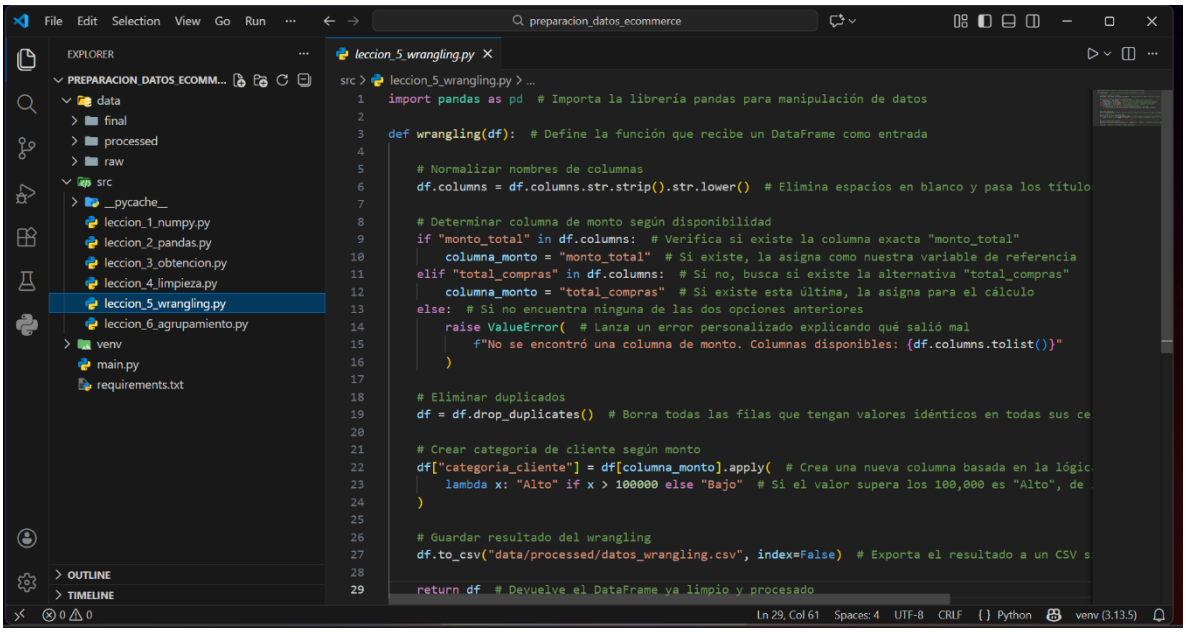

Imagen 5: Muestra la lección_5_wrangling.py

- Normalización: Estandarización de encabezados (minúsculas y sin espacios) para evitar fallos en el código.
- Segmentación: Creación de la columna categoria_cliente mediante funciones lambda para clasificar el valor del cliente ("Alto" > 100,000).

5. Agrupamiento y Resultados Finales (leccion_6_agrupamiento.py)

Para cumplir con la solicitud de la gerencia de contar con reportes de negocio, se realizaron:

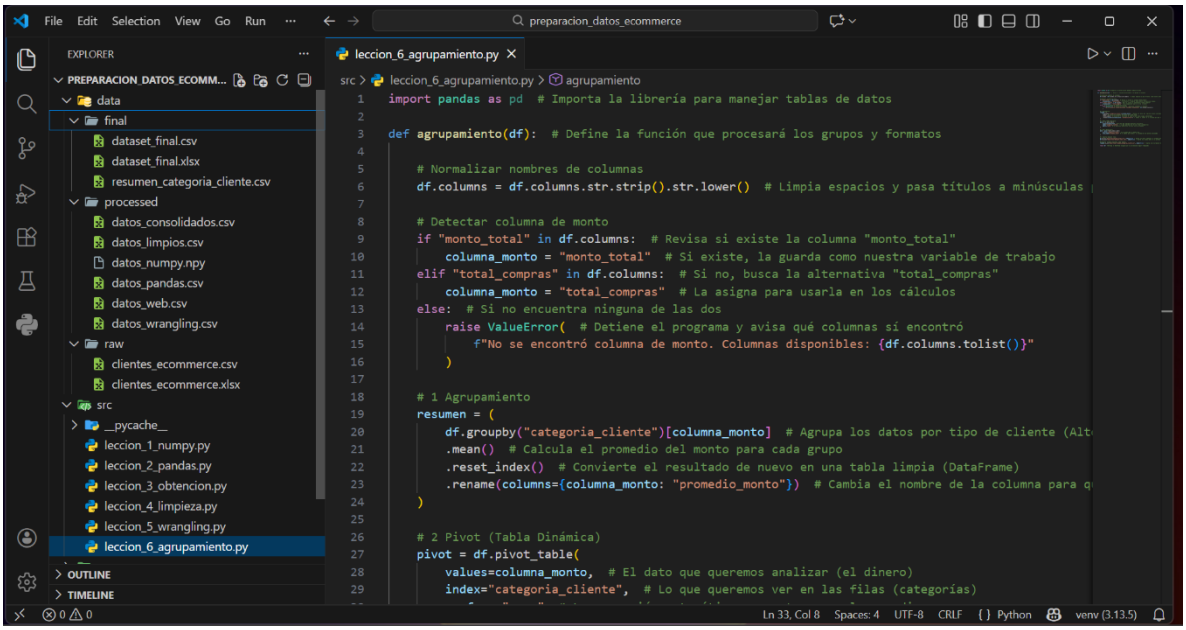

Imagen 6: Muestra la lección_6_agrupamiento.py

1. Agrupaciones: Promedio de gasto por categoría de cliente.
2. Tablas Dinámicas (Pivot): Resumen ejecutivo de los montos totales.
3. El uso de melt() fue para pasar de un formato "ancho" a uno "largo", lo cual es una mejor práctica para herramientas de visualización de datos.

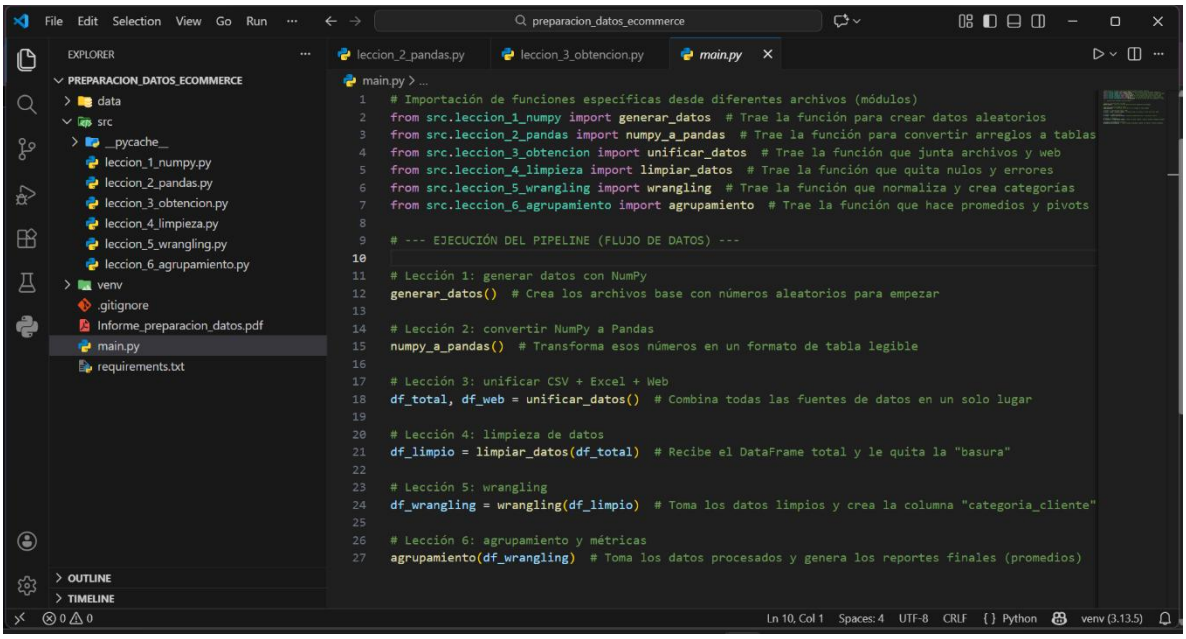

Imagen 7: Muestra el archivo main.py

main.py actúa como el orquestador o director de orquesta de todo el flujo de trabajo. Su función principal es ejecutar de manera secuencial y automática cada uno de los módulos que creaste en la carpeta src/, permitiendo que los datos fluyan desde su estado crudo hasta el reporte final.

6. Decisiones Tomadas y Desafíos Encontrados

1. Enfoque Modular: Se decidió separar el proyecto en 6 lecciones conectadas por un archivo main.py. Esto permite que si una fuente de datos cambia (por ejemplo, el Excel), solo se deba modificar un módulo sin romper todo el flujo.
2. Protección de Datos: Se trabajó con copias de los DataFrames (df.copy()) para asegurar que la información original nunca se vea alterada durante el proceso de limpieza.
3. Detección Dinámica de Columnas: Uno de los desafíos fue que las fuentes externas tenían nombres de columnas distintos ("monto_total" vs "total_compras"). Se programó una lógica de detección automática para que el código sea resiliente.
4. Inestabilidad de Fuentes Web: Las páginas web pueden bloquear el acceso automatizado o presentar errores de conexión inesperados. Para mitigar esto, el código debe estar preparado para manejar fallos si la URL no responde o si la estructura de la tabla HTML cambia sin previo aviso. (HTTP Error 403: Forbidden, indica que la página web está bloqueando accesos automáticos)

7. Estado Final del Dataset

El proyecto concluye con la generación de archivos en la ruta data/final/:

- dataset_final.csv/xlsx: Un archivo unificado, sin duplicados, con valores nulos corregidos y categorías asignadas.
- resumen_categoria_cliente.csv: Un reporte resumido

8. Conclusiones

I. Calidad y Fiabilidad de los Datos: La limpieza mediante técnicas estadísticas avanzadas garantiza que la toma de decisiones se base en información precisa. Al eliminar duplicados y corregir valores atípicos, se evitan errores en los reportes que podrían comprometer la estrategia comercial de la empresa.

II. Eficiencia mediante Arquitectura Modular: El diseño estructurado en bloques independientes bajo un archivo central permite que el flujo de datos sea escalable y fácil de mantener. Esta organización facilita la integración futura de nuevas fuentes de información sin necesidad de reescribir el sistema completo.

III. Versatilidad Técnica en la Transformación: El uso estratégico de herramientas como Lambda, Pivot y Melt permite adaptar el dataset a cualquier requerimiento técnico o de negocio. Estas funciones aseguran que la estructura final sea compatible con modelos de aprendizaje automático y herramientas de visualización.