

Informe de Proyecto:

Módulo 3: Obtención y Preparación de Datos

Nombre: Camila Garrido

1. Introducción

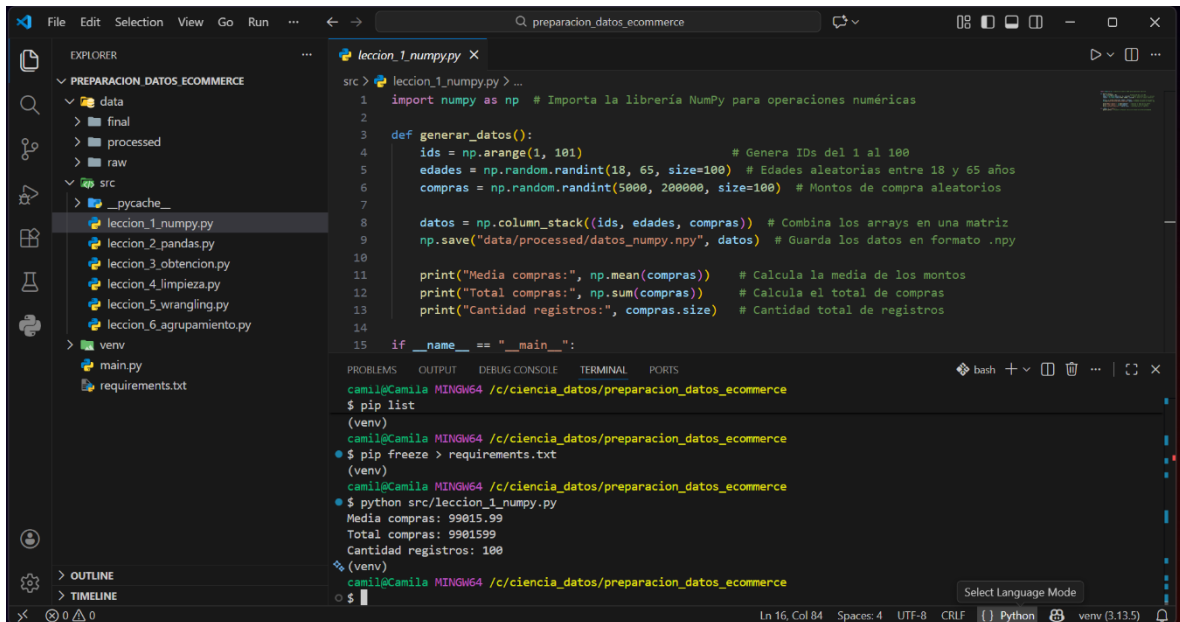
El presente proyecto responde a la necesidad del Equipo de Analítica de Datos de una empresa de e-commerce para resolver la falta de estructura y calidad en su información. Los datos originales presentaban valores nulos, duplicados y outliers.

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:



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure for 'PREPARACION_DATOS_ECOMMERCE' with folders 'data', 'final', 'processed', 'raw', and 'src'. The 'src' folder contains files like 'leccion_1_numpy.py', 'leccion_2_pandas.py', 'leccion_3_obtencion.py', 'leccion_4_limpieza.py', 'leccion_5_wrangling.py', and 'leccion_6_agrupamiento.py'. The 'leccion_1_numpy.py' file is open in the editor, showing the following code:

```
src > leccion_1_numpy.py > ...
1 import numpy as np # Importa la libreria NumPy para operaciones numéricas
2
3 def generar_datos():
4     ids = np.arange(1, 101) # Genera IDs del 1 al 100
5     edades = np.random.randint(18, 65, size=100) # Edades aleatorias entre 18 y 65 años
6     compras = np.random.randint(5000, 200000, size=100) # Montos de compra aleatorios
7
8     datos = np.column_stack((ids, edades, compras)) # Combina los arrays en una matriz
9     np.save("data/processed/datos_numpy.npy", datos) # Guarda los datos en formato .npy
10
11     print("Media compras:", np.mean(compras)) # Calcula la media de los montos
12     print("Total compras:", np.sum(compras)) # Calcula el total de compras
13     print("Cantidad registros:", compras.size) # Cantidad total de registros
14
15 if __name__ == "__main__":
16     generar_datos()
```

The terminal at the bottom shows the execution of the script:

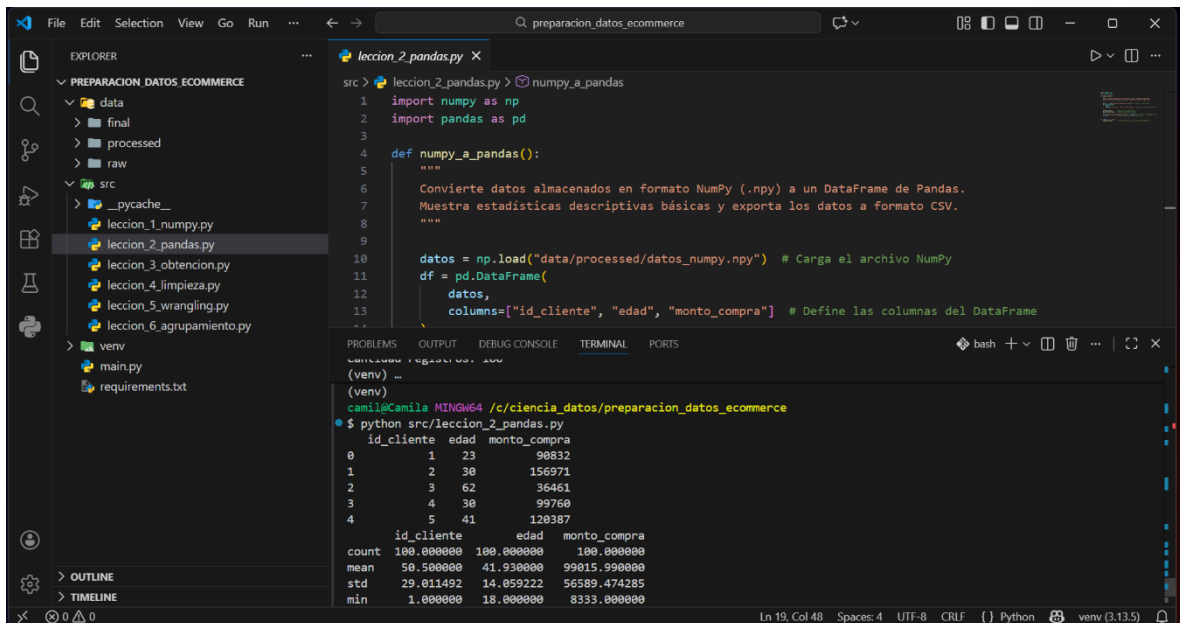
```
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$ pip list
(venv)
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$ pip freeze > requirements.txt
(venv)
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$ python src/leccion_1_numpy.py
Media compras: 99815.99
Total compras: 9981599
Cantidad registros: 100
(venv)
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$
```

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:



The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** A file tree on the left showing a project named 'PREPARACION_DATOS_ECOMMERCE'. It contains folders 'data' (with subfolders 'final', 'processed', 'raw') and 'src'. The 'src' folder contains several Python files, with 'leccion_2_pandas.py' selected.
- EDITOR:** The main window displays the code for 'leccion_2_pandas.py'. The code imports 'numpy' and 'pandas', defines a function 'numpy_a_pandas()' with a docstring, and then loads a NumPy file into a DataFrame with columns 'id_cliente', 'edad', and 'monto_compra'.
- TERMINAL:** The bottom panel shows the command prompt output. It displays the command to run the script and the resulting DataFrame, which includes individual data rows and summary statistics (count, mean, std, min).

```
src > leccion_2_pandas.py > numpy_a_pandas
1 import numpy as np
2 import pandas as pd
3
4 def numpy_a_pandas():
5     """
6     Convierte datos almacenados en formato NumPy (.npy) a un DataFrame de Pandas.
7     Muestra estadísticas descriptivas básicas y exporta los datos a formato CSV.
8     """
9
10    datos = np.load("data/processed/datos_numpy.npy") # Carga el archivo NumPy
11    df = pd.DataFrame(
12        datos,
13        columns=["id_cliente", "edad", "monto_compra"] # Define las columnas del DataFrame
14    )
15
```

```
(venv) ...
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$ python src/leccion_2_pandas.py
id_cliente  edad  monto_compra
0           1   23         98832
1           2   30        156971
2           3   62        36461
3           4   30        99760
4           5   41       128387

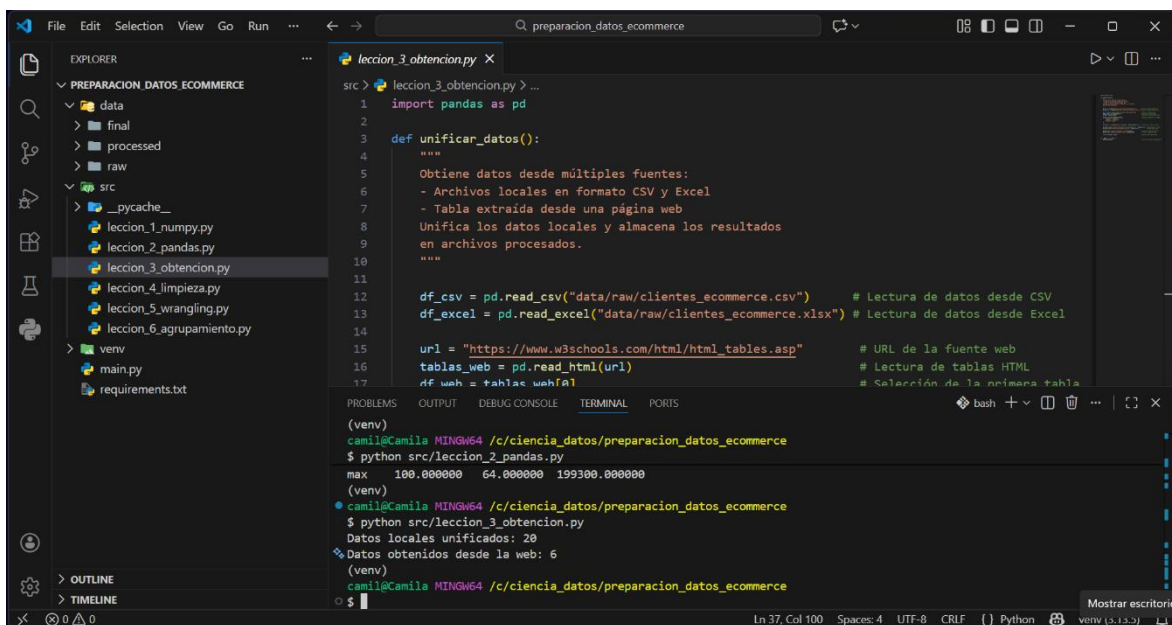
id_cliente  edad  monto_compra
count  100.000000  100.000000  100.000000
mean    50.500000  41.930000  99815.990000
std    29.011492  14.059222  56589.474285
min     1.000000  18.000000  8333.000000
```

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:



The screenshot shows a VS Code editor interface. On the left, the 'EXPLORER' sidebar displays a project structure for 'PREPARACION_DATOS_ECOMMERCE'. It includes folders 'data' (with subfolders 'final', 'processed', 'raw') and 'src'. The 'src' folder contains several Python files, with 'leccion_3_obtencion.py' selected. The main editor window shows the code for 'leccion_3_obtencion.py', which defines a function 'unificar_datos()' that reads data from CSV and Excel files and scrapes data from a website. The bottom panel shows the 'TERMINAL' with a bash shell. It displays the execution of 'python src/leccion_2_pandas.py' and 'python src/leccion_3_obtencion.py', showing output for data dimensions and the number of records obtained from local files and the web.

```
src > leccion_3_obtencion.py > ...
1 import pandas as pd
2
3 def unificar_datos():
4     """
5     Obtiene datos desde múltiples fuentes:
6     - Archivos locales en formato CSV y Excel
7     - Tabla extraída desde una página web
8     Unifica los datos locales y almacena los resultados
9     en archivos procesados.
10    """
11
12    df_csv = pd.read_csv("data/raw/clientes_ecommerce.csv") # Lectura de datos desde CSV
13    df_excel = pd.read_excel("data/raw/clientes_ecommerce.xlsx") # Lectura de datos desde Excel
14
15    url = "https://www.w3schools.com/html/html_tables.asp" # URL de la fuente web
16    tablas_web = pd.read_html(url) # Lectura de tablas HTML
17    df_web = tablas_web[0] # Selección de la primera tabla
```

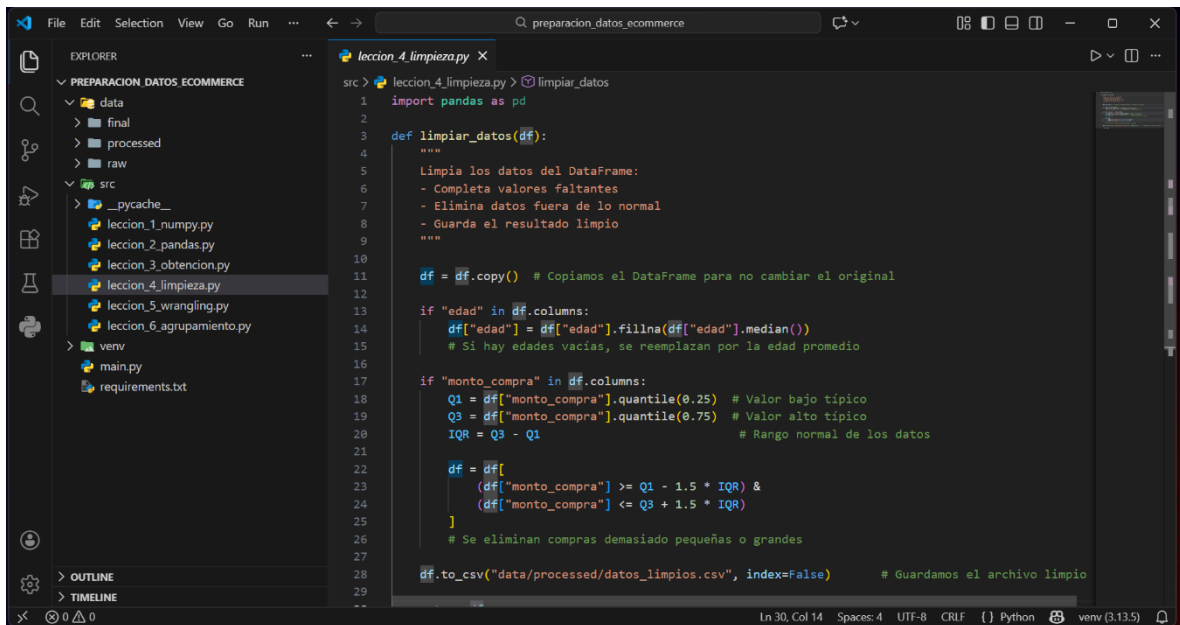
```
(venv)
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$ python src/leccion_2_pandas.py
max 100.000000 64.000000 199300.000000
(venv)
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$ python src/leccion_3_obtencion.py
Datos locales unificados: 20
Datos obtenidos desde la web: 6
(venv)
camil@Camila MINGW64 /c/ciencia_datos/preparacion_datos_ecommerce
$
```

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)



```
src > leccion_4_limpieza.py > limpiar_datos
1 import pandas as pd
2
3 def limpiar_datos(df):
4     """
5     Limpia los datos del DataFrame:
6     - Completa valores faltantes
7     - Elimina datos fuera de lo normal
8     - Guarda el resultado limpio
9     """
10
11     df = df.copy() # Copiamos el DataFrame para no cambiar el original
12
13     if "edad" in df.columns:
14         df["edad"] = df["edad"].fillna(df["edad"].median())
15         # Si hay edades vacías, se reemplazan por la edad promedio
16
17     if "monto_compra" in df.columns:
18         Q1 = df["monto_compra"].quantile(0.25) # Valor bajo típico
19         Q3 = df["monto_compra"].quantile(0.75) # Valor alto típico
20         IQR = Q3 - Q1 # Rango normal de los datos
21
22         df = df[
23             (df["monto_compra"] >= Q1 - 1.5 * IQR) &
24             (df["monto_compra"] <= Q3 + 1.5 * IQR)
25         ]
26         # Se eliminan compras demasiado pequeñas o grandes
27
28     df.to_csv("data/processed/datos_limpios.csv", index=False) # Guardamos el archivo limpio
29
30
```

Imagen 4: Muestra la lección_4_limpieza.py

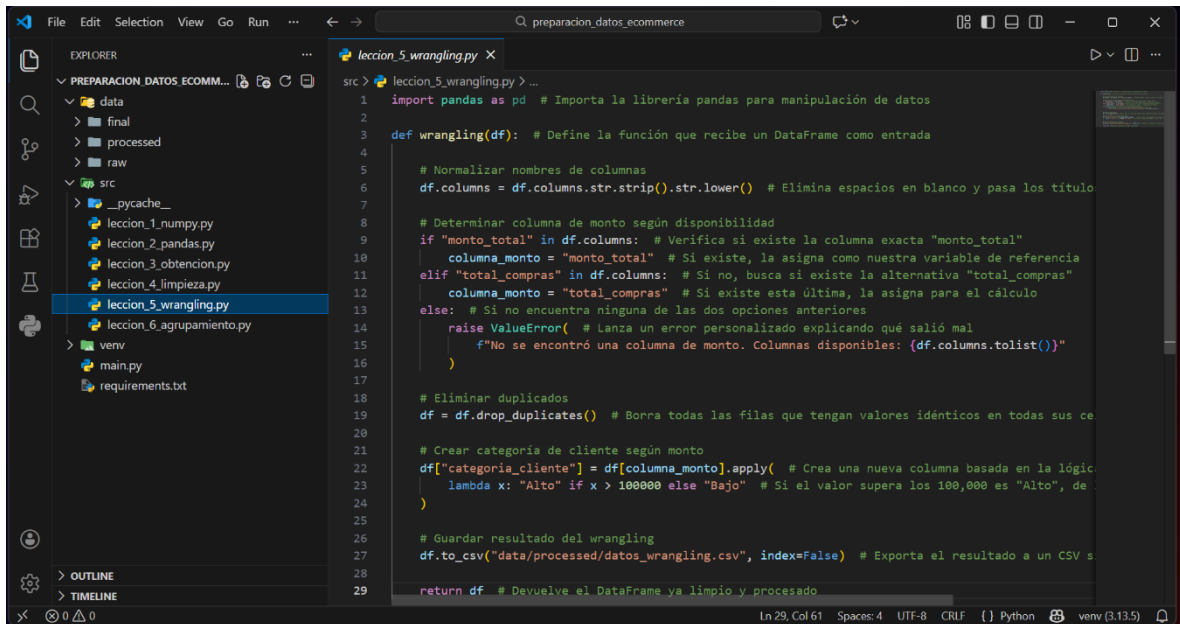
- Tratamiento de Nulos: Se aplicó la técnica de imputación por mediana en la columna edad. Se eligió la mediana sobre el promedio porque es más robusta frente a valores extremos.
- Detección de Outliers: Se utilizó el Rango Intercuartílico (IQR):

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)



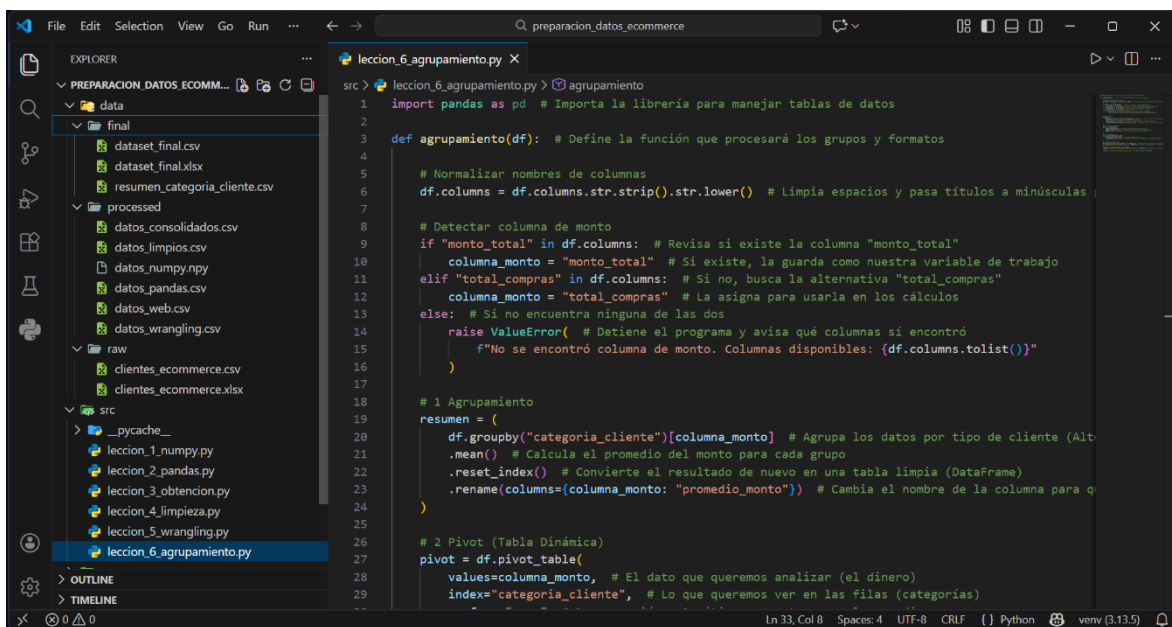
```
src > leccion_5_wrangling.py > ...
1 import pandas as pd # Importa la libreria pandas para manipulaci3n de datos
2
3 def wrangling(df): # Define la funci3n que recibe un DataFrame como entrada
4
5     # Normalizar nombres de columnas
6     df.columns = df.columns.str.strip().str.lower() # Elimina espacios en blanco y pasa los t3tulo
7
8     # Determinar columna de monto seg3n disponibilidad
9     if "monto_total" in df.columns: # Verifica si existe la columna exacta "monto_total"
10         columna_monto = "monto_total" # Si existe, la asigna como nuestra variable de referencia
11     elif "total_compras" in df.columns: # Si no, busca si existe la alternativa "total_compras"
12         columna_monto = "total_compras" # Si existe esta 3ltima, la asigna para el c3lculo
13     else: # Si no encuentra ninguna de las dos opciones anteriores
14         raise ValueError( # Lanza un error personalizado explicando qu3 sali3 mal
15             f"No se encontr3 una columna de monto. Columnas disponibles: {df.columns.tolist()}"
16         )
17
18     # Eliminar duplicados
19     df = df.drop_duplicates() # Borra todas las filas que tengan valores id3nticos en todas sus ce
20
21     # Crear categor3a de cliente seg3n monto
22     df["categoria_cliente"] = df[columna_monto].apply( # Crea una nueva columna basada en la l3gic
23         lambda x: "Alto" if x > 100000 else "Bajo" # Si el valor supera los 100,000 es "Alto", de
24     )
25
26     # Guardar resultado del wrangling
27     df.to_csv("data/processed/datos_wrangling.csv", index=False) # Exporta el resultado a un CSV s
28
29     return df # Devuelve el DataFrame ya limpio y procesado
```

Imagen 5: Muestra la lecci3n_5_wrangling.py

- Normalizaci3n: Estandarizaci3n de encabezados (min3sculas y sin espacios) para evitar fallos en el c3digo.
- Segmentaci3n: Creaci3n 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:



```
src > leccion_6_agrupamiento.py > agrupamiento
1 import pandas as pd # Importa la librería para manejar tablas de datos
2
3 def agrupamiento(df): # Define la función que procesará los grupos y formatos
4
5     # Normalizar nombres de columnas
6     df.columns = df.columns.str.strip().str.lower() # Limpia espacios y pasa títulos a minúsculas
7
8     # Detectar columna de monto
9     if "monto_total" in df.columns: # Revisa si existe la columna "monto_total"
10         columna_monto = "monto_total" # Si existe, la guarda como nuestra variable de trabajo
11     elif "total_compras" in df.columns: # Si no, busca la alternativa "total_compras"
12         columna_monto = "total_compras" # La asigna para usarla en los cálculos
13     else: # Si no encuentra ninguna de las dos
14         raise ValueError( # Detiene el programa y avisa qué columnas sí encontró
15             f"No se encontró columna de monto. Columnas disponibles: {df.columns.tolist()}"
16         )
17
18     # 1 Agrupamiento
19     resumen = (
20         df.groupby("categoria_cliente")[columna_monto] # Agrupa los datos por tipo de cliente (Alt
21         .mean() # Calcula el promedio del monto para cada grupo
22         .reset_index() # Convierte el resultado de nuevo en una tabla limpia (DataFrame)
23         .rename(columns={columna_monto: "promedio_monto"}) # Cambia el nombre de la columna para q
24     )
25
26     # 2 Pivot (Tabla Dinámica)
27     pivot = df.pivot_table(
28         values=columna_monto, # El dato que queremos analizar (el dinero)
29         index="categoria_cliente", # Lo que queremos ver en las filas (categorías)
30         columns="promedio_monto"
31     )
```

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.

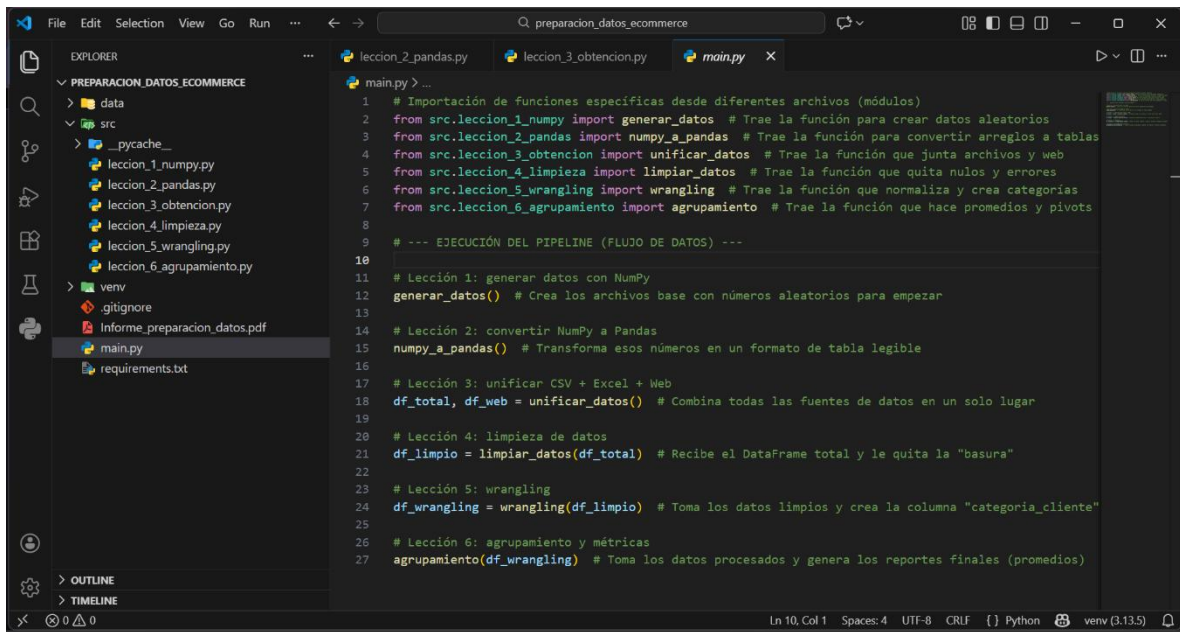


Imagen 7: Muestra el archivo main.py

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.