

# **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? (lección\_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 Jupyter Notebook interface with the following details:

- File Explorer:** Shows a project structure under "PREPARACION DATOS ECOMMERCE". The "src" folder contains several Python files: lección\_1\_numpy.py, lección\_2\_pandas.py, lección\_3\_obtención.py, lección\_4\_limpieza.py, lección\_5\_wrangling.py, and lección\_6\_agrupamiento.py.
- Code Cell:** Displays the content of "lección\_1\_numpy.py". The code generates synthetic data (IDs, ages, purchases), combines them into a matrix, and calculates mean, total, and count.
- Terminal:** Shows the command-line output of running the script. It includes pip installations, freezing requirements, and executing the script to print summary statistics.
- Status Bar:** Shows the current language mode as Python, the file path as "/c/ciencia\_datos/preparacion\_datos\_ecommerce", and the code editor version as 3.13.5.

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? (lección\_2\_pandas.py en adelante)

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

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows the project structure under "PREPARACION\_DATOS\_ECOMMERCE".
- Code Cell:** Contains the Python script "lección\_2\_pandas.py".
- Output:** Displays the execution results of the script, including the printed DataFrame and its summary statistics.
- Terminal:** Shows the command used to run the script: \$ python src/lección\_2\_pandas.py.
- Data Preview:** Shows the first few rows of the DataFrame.
- Summary Statistics:** Shows the count, mean, std, and min values for each column.

	id_cliente	edad	monto_compra
count	100.000000	100.000000	100.000000
mean	56.500000	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 Jupyter Notebook environment with the following details:

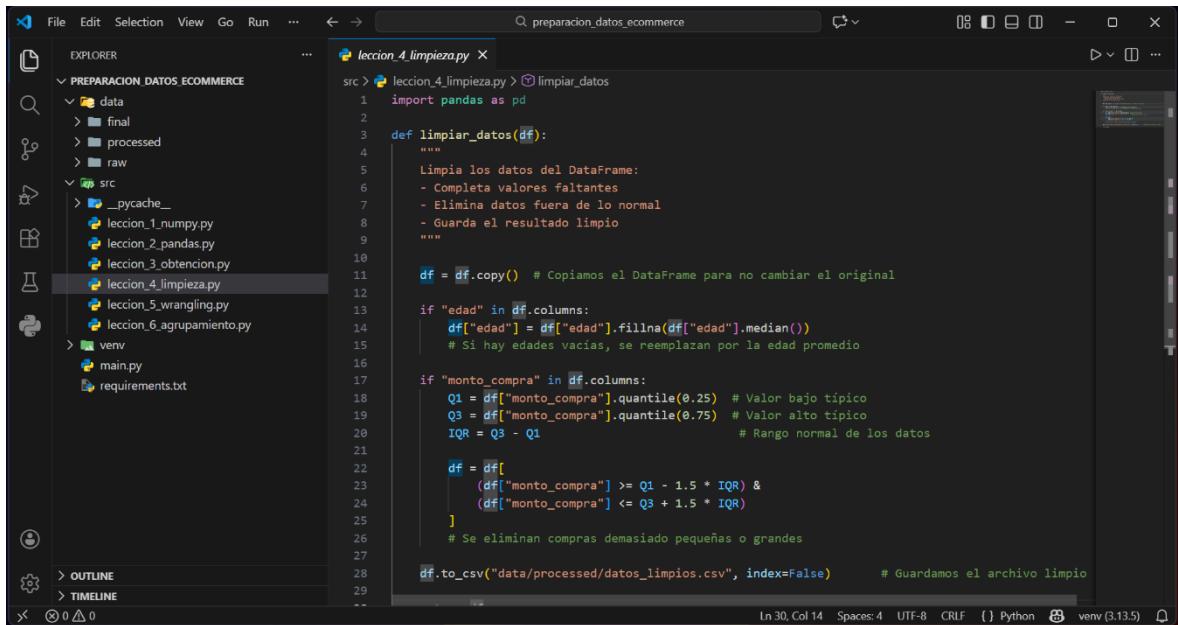
- File Explorer:** On the left, it lists the project structure under "PREPARACION\_DATOS\_ECOMMERCE". The "src" folder contains several Python files: leccion\_1\_numpy.py, leccion\_2\_pandas.py, leccion\_3\_obtencion.py (which is currently selected), leccion\_4\_limpieza.py, leccion\_5\_wrangling.py, and leccion\_6\_agrupamiento.py. It also includes venv, main.py, and requirements.txt.
- Code Editor:** The main area displays the content of "leccion\_3\_obtencion.py". The code imports pandas, defines a function to unify data from multiple sources, and reads data from CSV, Excel, and a web URL. It then concatenates the data and saves it to processed files.
- Terminal:** At the bottom, the terminal window shows the execution of the script. It starts with activating the virtual environment (venv) and running "python src/leccion\_2\_pandas.py". Then, it runs "python src/leccion\_3\_obtencion.py", which prints "Datos locales unificados: 20". Finally, it runs "python src/leccion\_4\_limpieza.py", which prints "Datos obtenidos desde la web: 6".

### **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 (lección\_4\_limpieza.py)



The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows a project structure under 'PREPARACION\_DATOS\_ECOMMERCE'. The 'src' directory contains several Python files: 'lección\_4\_limpieza.py', 'limpiar\_datos.py', 'lección\_1\_numpy.py', 'lección\_2\_pandas.py', 'lección\_3\_obtencion.py', 'lección\_4\_limpieza.py' (selected), 'lección\_5\_wrangling.py', and 'lección\_6\_agrupamiento.py'. It also includes 'venv', 'main.py', and 'requirements.txt'.
- Code Editor:** The main pane displays the content of 'lección\_4\_limpieza.py'. The code defines a function 'limpiar\_datos(df)' that handles missing values and removes outliers from the 'edad' and 'monto\_compra' columns. It uses Pandas' quantile method to calculate Q1 and Q3, and then filters rows where the values fall outside the range of Q1 - 1.5 \* IQR to Q3 + 1.5 \* IQR. Finally, it saves the cleaned data to a CSV file.
- Bottom Status Bar:** Shows the current line (Ln 30), column (Col 14), spaces (Spaces: 4), encoding (UTF-8), and file type (Python). It also indicates the environment ('venv (3.13.5)').

Imagen 4: Muestra la lección\_4\_limpieza.py

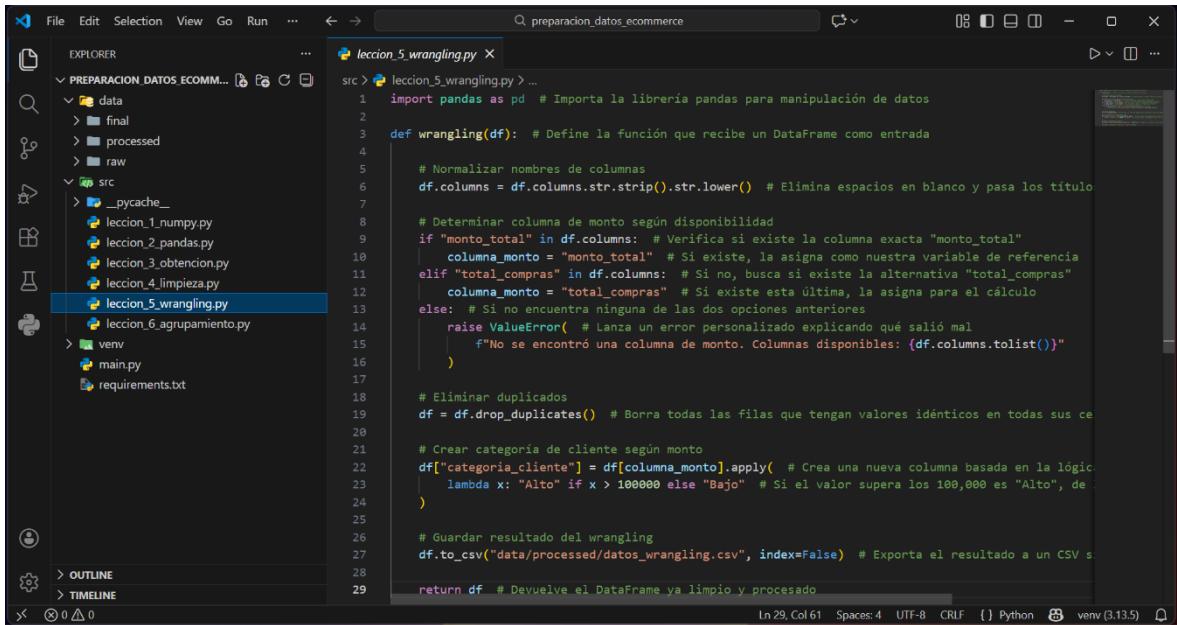
- Tratamiento de Nulos: Se aplicó la técnica de imputación por mediana en la columna edad. Se eligió la mediana sobre el promedio porque es más robusta frente a valores extremos.
- Detección de Outliers: Se utilizó el Rango Intercuartílico (IQR):

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 (lección\_5\_wrangling.py)



```
src > lección_5_wrangling.py > ...
1 import pandas as pd # Importa la librería pandas para manipulación de datos
2
3 def wrangling(df): # Define la función 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 titulos
7
8     # Determinar columna de monto según 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 última, la asigna para el cálculo
13    else: # Si no encuentra ninguna de las dos opciones anteriores
14        raise ValueError( # Lanza un error personalizado explicando qué salió mal
15            f"No se encontró 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 idénticos en todas sus columnas
20
21    # Crear categoría de cliente según monto
22    df["categoria_cliente"] = df[columna_monto].apply( # Crea una nueva columna basada en la lógica
23        lambda x: "Alto" if x > 100000 else "Bajo" # Si el valor supera los 100,000 es "Alto", de lo contrario es "Bajo"
24    )
25
26    # Guardar resultado del wrangling
27    df.to_csv("data/processed/datos_wrangling.csv", index=False) # Exporta el resultado a un CSV sin indices
28
29 return df # Devuelve el DataFrame ya limpio y procesado
```

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 (lección\_6\_agrupamiento.py)

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

```
File Edit Selection View Go Run ... < > preparacion_datos_comercio lección_6_agrupamiento.py agrupamiento
src > lección_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)
```

## 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.

```
File Edit Selection View Go Run ... ← → Q preparacion_datos_ecommerce
EXPLORER ... leccion_2_pandas.py leccion_3_obtencion.py main.py ...
PREPARACION_DATOS_ECOMMERCE
> data
src
> pycache_
    leccion_1_numpy.py
    leccion_2_pandas.py
    leccion_3_obtencion.py
    leccion_4_limpieza.py
    leccion_5_wrangling.py
    leccion_6_agrupamiento.py
venv
gitignore
Informe_preparacion_datos.pdf
main.py
requirements.txt

main.py > ...
1 # Importación de funciones específicas desde diferentes archivos (módulos)
2 from src.leccion_1_numpy import generar_datos # Trae la función para crear datos aleatorios
3 from src.leccion_2_pandas import numpy_a_pandas # Trae la función para convertir arreglos a tablas
4 from src.leccion_3_obtencion import unificar_datos # Trae la función que junta archivos y web
5 from src.leccion_4_limpieza import limpiar_datos # Trae la función que quita nulos y errores
6 from src.leccion_5_wrangling import wrangling # Trae la función que normaliza y crea categorías
7 from src.leccion_6_agrupamiento import agrupamiento # Trae la función que hace promedios y pivots
8
9 # --- EJECUCIÓN DEL PIPELINE (FLUJO DE DATOS) ---
10
11 # Lección 1: generar datos con NumPy
12 generar_datos() # Crea los archivos base con números aleatorios para empezar
13
14 # Lección 2: convertir NumPy a Pandas
15 numpy_a_pandas() # Transforma esos números en un formato de tabla legible
16
17 # Lección 3: unificar CSV + Excel + Web
18 df_total, df_web = unificar_datos() # Combina todas las fuentes de datos en un solo lugar
19
20 # Lección 4: limpieza de datos
21 df_limpio = limpiar_datos(df_total) # Recibe el DataFrame total y le quita la "basura"
22
23 # Lección 5: wrangling
24 df_wrangling = wrangling(df_limpio) # Toma los datos limpios y crea la columna "categoria_cliente"
25
26 # Lección 6: agrupamiento y métricas
27 agrupamiento(df_wrangling) # Toma los datos procesados y genera los reportes finales (promedios)

Ln 10, Col 1 Spaces:4 UTF-8 CRLF {} Python venv (3.13.5)
```

**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.