Proyecto final: análisis completo de los pedidos de PyShop

Durante el curso trabajaste con el CSV de pedidos 2025 que juntaron las sucursales de PyShop: lo exploraste, lo limpiaste, trataste sus faltantes, lo agrupaste y lo graficaste. Ahora llega un archivo nuevo, más grande, con los mismos defectos en otra proporción. Tu tarea es entregar un análisis completo y reproducible: limpieza validada, reporte de calidad, respuestas a cinco preguntas de negocio, gráficas y conclusiones. Se usan Python 3.11 o superior y pandas (el curso usa pandas 3); para las gráficas basta con `.plot`. El notebook `proyecto-pedidos` contiene una solución de referencia: consúltalo al terminar, no antes.

Contexto

La dirección de PyShop recibió un nuevo consolidado de pedidos y quiere saber dos cosas: si los datos son confiables y qué dicen sobre categorías, ciudades y productos. No le interesa el código, sino un reporte claro; pero el análisis debe poder repetirse con el archivo del próximo periodo sin cambiar nada.

Datos

Usa la función `generar_csv_pedidos`, que está en la primera celda de código de cualquier notebook del curso, con **otros argumentos**: 1500 pedidos y semilla 2026. Devuelve el texto de un CSV con las mismas diez columnas de siempre.

```
import io
import pandas as pd

texto = generar_csv_pedidos(n=1500, semilla=2026)
crudo = pd.read_csv(io.StringIO(texto))
print(crudo.shape)   # (1526, 10)
```

Si tu `shape` no es `(1526, 10)`, revisa los argumentos de la función. Todas las demás cifras debes obtenerlas tú.

Requisitos mínimos

1. **Reporte de calidad.** Una función `reporte_calidad(df)` que devuelva, al menos: filas, filas duplicadas, ciudades distintas, tipo del precio, nulos por columna y fechas que no se pueden convertir. Aplícala al crudo y al limpio, y muestra ambas en una tabla «antes / después».

2. **Limpieza.** Una función `limpiar(texto)` que devuelva un DataFrame limpio: sin duplicados, cinco ciudades bien escritas, precio numérico, fecha como `datetime64`, faltantes de cantidad y precio tratados con un criterio justificado, filas sin fecha válida descartadas y una columna `total`. Se recomienda escribirla como cadena de métodos con `assign`.
3. **Validación.** Una función `validar(df)` con `assert` que compruebe, al menos, que `pedido_id` es único, que hay exactamente cinco ciudades sin nulos y que cantidad, precio, fecha y total no tienen nulos.
4. **La calificación.** Decide qué hacer con sus faltantes y explica por qué en una celda de texto.
5. **Cinco preguntas de negocio**, sin contar los pedidos cancelados en el ingreso:
   - P1. ¿Qué categoría genera más ingreso y qué porcentaje del total aporta cada una?
   - P2. Por ciudad: pedidos, ingreso, ticket medio y calificación media (con `agg` con nombres). ¿Qué ciudad tiene el ticket medio más alto?
   - P3. Ingreso por categoría y ciudad con `pivot_table`.
   - P4. Tasa de cancelación por ciudad, en porcentaje (aquí sí cuentan los cancelados).
   - P5. Los tres productos con más unidades vendidas.
6. **Tres gráficas** con `.plot`: un ranking (barras ordenadas), una tendencia mensual (línea con el eje desde cero) y una tercera a tu elección. Cada una con título y unidades.
7. **Conclusiones:** entre cuatro y seis frases, cada una con una cifra, y al menos una sobre lo que los datos *no* muestran.

Pistas de implementación

Revisa las variantes de ciudad antes de escribir el diccionario:

```
base = crudo["ciudad"].str.strip().str.lower()
print(sorted(base.unique()))
```

Dentro de una cadena, usa `lambda d:` para que cada columna se calcule sobre el DataFrame ya sin duplicados:

```
limpio = crudo.drop_duplicates().assign(
    fecha=lambda d: pd.to_datetime(
        d["fecha"], errors="coerce"),
)
```

La tasa de cancelación es la media de una columna booleana:

```
cancelado = pedidos["estado"] == "cancelado"
tasa = cancelado.groupby(pedidos["ciudad"]).mean()
print((tasa * 100).round(1))
```

El generador de datos no modela estacionalidad ni diferencias reales entre ciudades: si una cifra sube o baja, descríbela, pero no le inventes una causa.

Extras opcionales

- **Precios contra el catálogo.** Crea un DataFrame con `PRODUCTOS` y únelo a los pedidos con `merge(on="producto", validate="many_to_one")`. Compara el precio del archivo (convertido a número, antes de rellenar) con el del catálogo e informa cuántos difieren. ¿Qué te dice el resultado sobre los 33 precios distintos del crudo?
- **Exportar a Excel.** Con `pd.ExcelWriter` guarda en un solo archivo una hoja con los pedidos limpios y otra por cada tabla de P2 y P3. Requiere `pip install openpyxl`.
- **Memoria.** Convierte `estado` y `ciudad` a `category` y mide el ahorro con `memory_usage(deep=True, index=False)`.
- **Una sola figura.** Reúne las tres gráficas en `plt.subplots(1, 3)` con `ax=` y guárdala con `savefig`.

Entregables

1. **Un notebook** que corra de principio a fin sin errores: datos, reporte de calidad, `limpiar`, `validar`, preguntas, gráficas y conclusiones.
2. **El CSV limpio**, `pedidos_limpios.csv`, guardado con `index=False`.
3. **Las conclusiones** en una celda de texto al final del notebook.

Rúbrica (100 puntos, más 10 de extras)

- **Reporte de calidad (15 puntos):** mide lo pedido, antes y después, en una tabla.
- **Limpieza (25 puntos):** función reutilizable, orden correcto de los pasos y criterios de relleno justificados.
- **Validación (10 puntos):** los `assert` pedidos y al menos uno propio.
- **Preguntas de negocio (25 puntos):** cinco respuestas correctas, sin cancelados en el ingreso y con la tasa calculada sobre todos los pedidos.
- **Gráficas (10 puntos):** tipo adecuado, ordenadas, con título y unidades.
- **Conclusiones (15 puntos):** cifras concretas, una limitación de los datos y ninguna causa inventada.