qdagjjole

November 10, 2025

1 1. Importación y normalización

1.1 1.1. Carga de los archivos

[47]:
```
import pandas as pd
ventas = pd.read_csv("../data/ventas.csv")
productos = pd.read_csv("../data/productos.csv")
inventario = pd.read_csv("../data/inventario.csv")
costos = pd.read_csv("../data/Costos_Logisticos.csv")
```

1.2 1.2. Homogenización de nombre

[48]:
```
for df in [ventas, productos, inventario, costos]:
    df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")
```

1.3 1.3. Comprobación de la estructura

[49]:
```
for name, df in [("ventas", ventas), ("productos", productos),
                 ("inventario", inventario), ("costos", costos)]:
    print(f"\n{name}: {df.shape} filas × {df.shape[1]} columnas")
    display(df.head(3))
```

```
ventas: (940550, 7) filas × 7 columnas
```

	fecha	tienda	sku	cantidad_vendida	precio_venta_unitario	descuento_aplicado	venta_neta
0	2023-01-01	TIENDA001	HM000088	1	28.24	0.2	22.592
1	2023-01-01	TIENDA001	HM000096	1	64.60	0.1	58.140
2	2023-01-01	TIENDA001	HM000100	2	28.23	0.2	45.168

```
productos: (100, 13) filas × 13 columnas
```

```
        sku  categoria subcategoria talla   color  costo_unitario  \
0  HM000001   Vestidos    Temporada    XL   Negro           10.35
1  HM000002  Camisetas    Temporada     L    Rojo           34.60
2  HM000003    Abrigos       Basico     L  Blanco           40.50

   precio_venta proveedor_id  lead_time_dias  cantidad_minima_pedido  \
0         17.75      PROV001              18                      50
1         40.98      PROV002              12                      50
2         68.38      PROV003              12                     100

   multiplico_pedido temporada  margen
0                 10  Invierno    7.40
1                 10    Verano    6.38
2                 25    Verano   27.88


inventario: (600, 6) filas × 6 columnas

      tienda       sku  stock_actual  stock_seguridad_actual  \
0  TIENDA001  HM000084            18                      16
1  TIENDA001  HM000054            54                      23
2  TIENDA001  HM000071            62                      17

   stock_en_transito fecha_ultima_reposicion
0                 16              2024-01-06
1                 20              2023-12-30
2                 26              2024-01-03


costos: (5, 7) filas × 7 columnas

     escenario  costo_pedido  costo_mantenimiento_anual  \
0  Escenario 1          10.0                       0.15
1  Escenario 2          40.0                       0.20
2  Escenario 3          50.0                       0.25

   costo_almacenamiento_m2  costo_transferencia_tienda  costo_rotura_stock  \
0                     80.0                         3.0                0.20
1                    100.0                         4.0                0.25
2                    120.0                         5.0                0.30

   costo_obsolescencia
0                 0.09
1                 0.12
2                 0.15
```

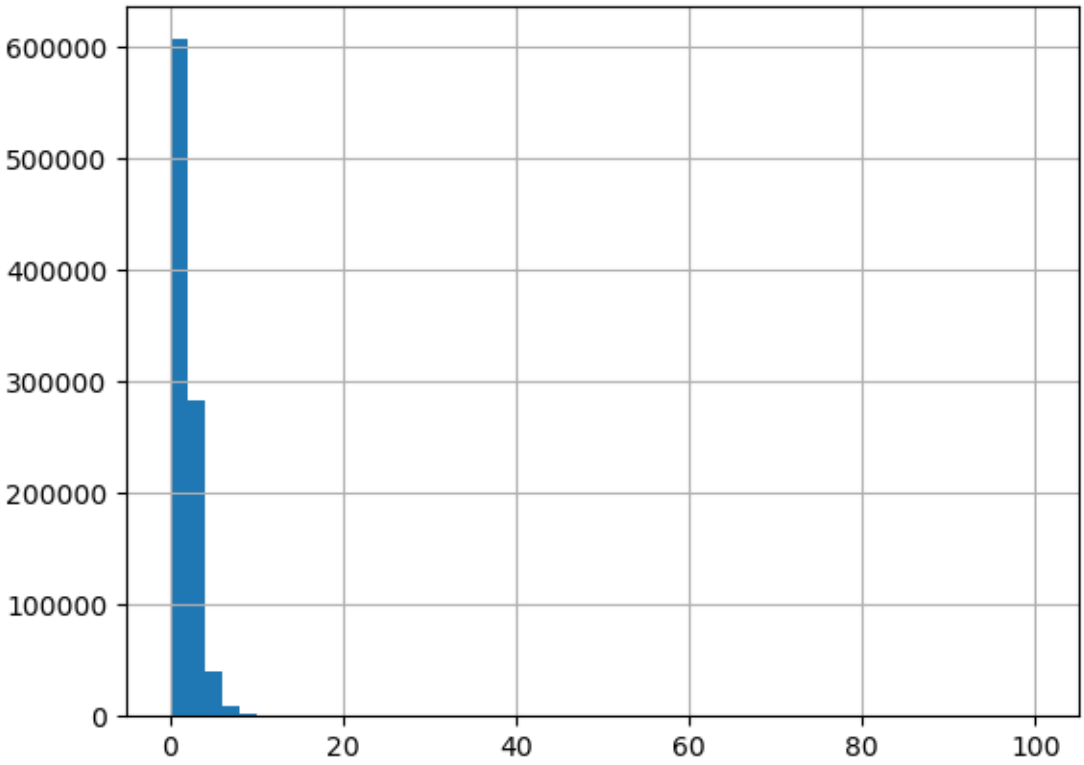
2 2.Validación de tipos y limpieza

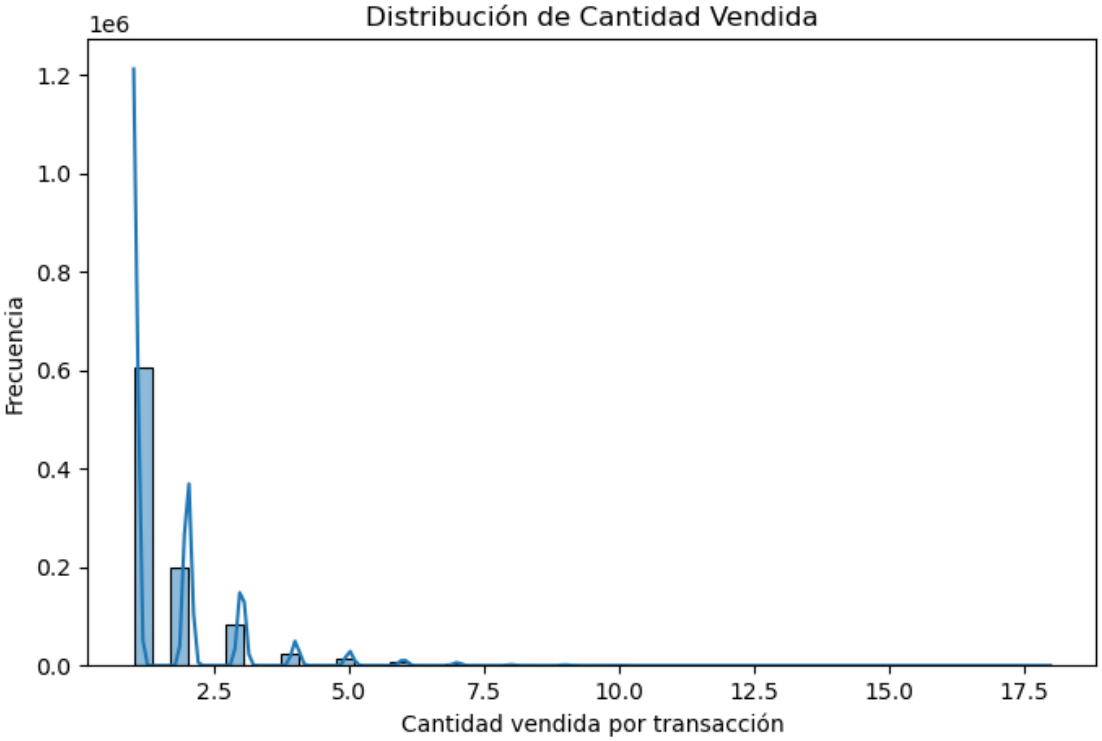
[50]:
```
ventas['fecha'] = pd.to_datetime(ventas['fecha'])
cols_num = ['cantidad_vendida','precio_venta_unitario','venta_neta']
ventas[cols_num] = ventas[cols_num].apply(pd.to_numeric, errors='coerce')
#Revisión nulos y duplicados
ventas.isna().sum(), ventas.duplicated().sum()
```

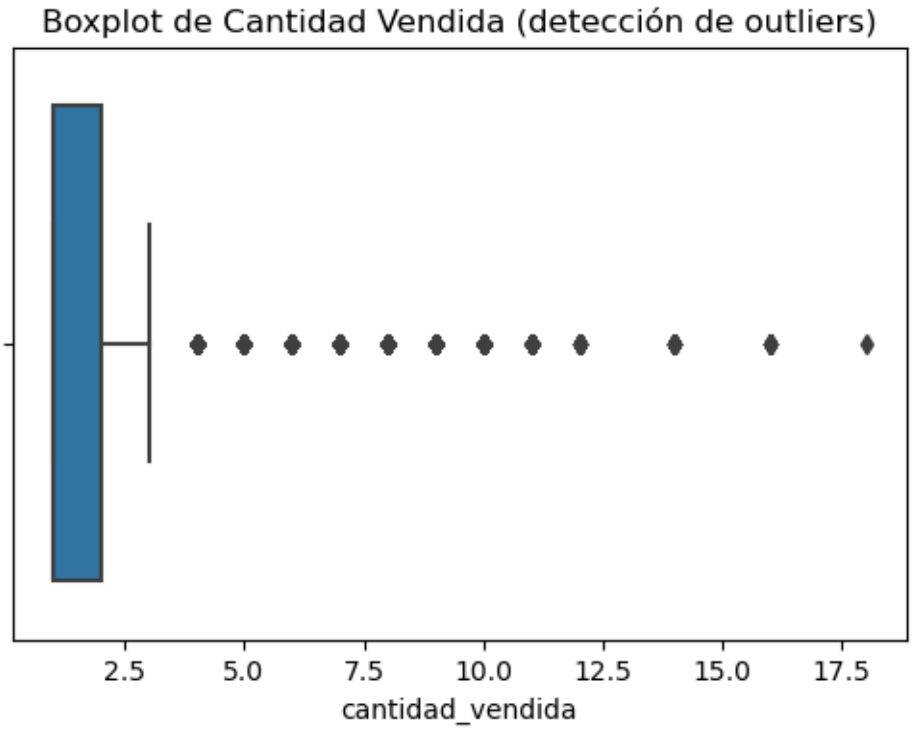
[50]:
```
(fecha                   0
 tienda                  0
 sku                     0
 cantidad_vendida        0
 precio_venta_unitario   0
 descuento_aplicado      0
 venta_neta              0
 dtype: int64,
 206573)
```

3 3. Descripción estadística inicial

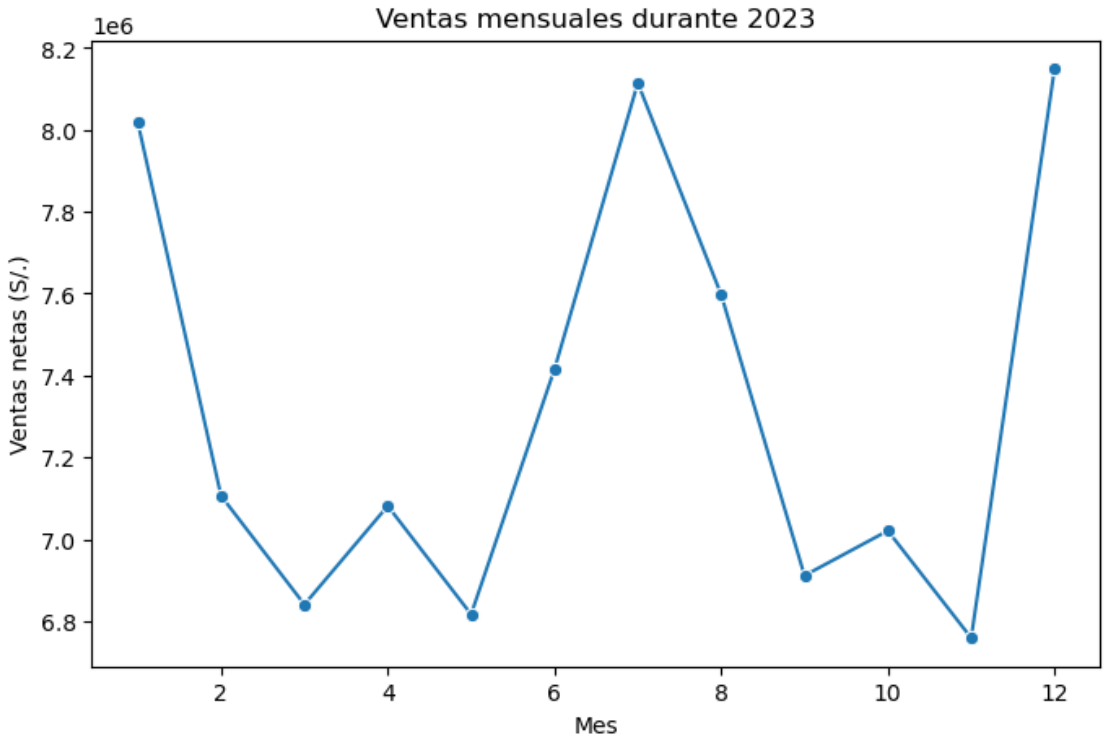
[51]:
```
import matplotlib.pyplot as plt
import seaborn as sns
ventas.describe(include='all', datetime_is_numeric=True)
ventas['cantidad_vendida'].hist(bins=50, range=(0, 100))
ventas[['precio_venta_unitario','cantidad_vendida']].corr()
#Tabla de resumen concentrado
resumen = ventas[['precio_venta_unitario','cantidad_vendida']].agg(
    ['count','mean','std','min','median','max']
)
resumen
#Histograma y distribución de ventas
plt.figure(figsize=(8,5))
sns.histplot(ventas['cantidad_vendida'], bins=50, kde=True)
plt.title("Distribución de Cantidad Vendida")
plt.xlabel("Cantidad vendida por transacción")
plt.ylabel("Frecuencia")
plt.show()
#Boxplots para identificar outliers
plt.figure(figsize=(6,4))
sns.boxplot(x=ventas['cantidad_vendida'])
plt.title("Boxplot de Cantidad Vendida (detección de outliers)")
plt.show()
```


600000
500000
400000
300000
200000
100000
0
0
20
40
60
80
100


Distribución de Cantidad Vendida
1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Frecuencia
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Cantidad vendida por transacción



[52]:
```
#Análisis temporal (por mes)
ventas['mes'] = ventas['fecha'].dt.month
ventas_mensual = ventas.groupby('mes')['venta_neta'].sum().reset_index()

plt.figure(figsize=(8,5))
sns.lineplot(data=ventas_mensual, x='mes', y='venta_neta', marker='o')
plt.title("Ventas mensuales durante 2023")
plt.xlabel("Mes")
plt.ylabel("Ventas netas (S/.)")
plt.show()
```



[53]:
```
print("VENTAS:", ventas.columns.tolist())
print("PRODUCTOS:", productos.columns.tolist())
print("INVENTARIO:", inventario.columns.tolist())
```

```
VENTAS: ['fecha', 'tienda', 'sku', 'cantidad_vendida', 'precio_venta_unitario',
'descuento_aplicado', 'venta_neta', 'mes']
PRODUCTOS: ['sku', 'categoria', 'subcategoria', 'talla', 'color',
'costo_unitario', 'precio_venta', 'proveedor_id', 'lead_time_dias',
'cantidad_minima_pedido', 'multiplico_pedido', 'temporada', 'margen']
INVENTARIO: ['tienda', 'sku', 'stock_actual', 'stock_seguridad_actual',
'stock_en_transito', 'fecha_ultima_reposicion']
```

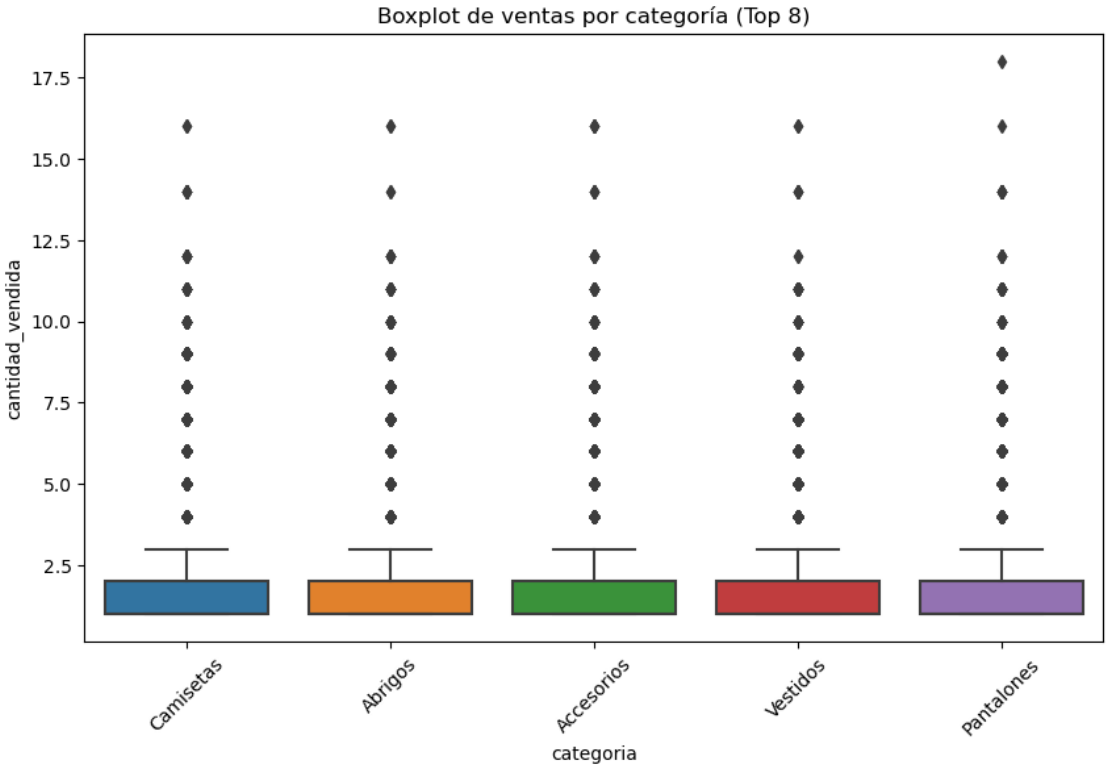
[54]:
```
#Uniendo ventas con productos
ventas_cat = ventas.merge(productos[['sku','categoria']], on='sku', how='left')
ventas_cat.head()
```

[54]:

	fecha	tienda	sku	cantidad_vendida	precio_venta_unitario	\
0	2023-01-01	TIENDA001	HM000088	1	28.24	
1	2023-01-01	TIENDA001	HM000096	1	64.60	
2	2023-01-01	TIENDA001	HM000100	2	28.23	
3	2023-01-01	TIENDA001	HM000088	6	28.24	
4	2023-01-01	TIENDA001	HM000084	2	93.95	

	descuento_aplicado	venta_neta	mes	categoria
0	0.2	22.592	1	Camisetas
1	0.1	58.140	1	Abrigos
2	0.2	45.168	1	Accesorios
3	0.0	169.440	1	Camisetas
4	0.1	169.110	1	Accesorios

[55]:
```
import seaborn as sns
import matplotlib.pyplot as plt

top_categorias = ventas_cat['categoria'].value_counts().nlargest(8).index
ventas_top = ventas_cat[ventas_cat['categoria'].isin(top_categorias)]

plt.figure(figsize=(10,6))
sns.boxplot(data=ventas_top, x='categoria', y='cantidad_vendida')
plt.xticks(rotation=45)
plt.title("Boxplot de ventas por categoría (Top 8)")
plt.show()
```



[56]:
```
print(ventas_cat.columns.tolist())
ventas_cat[['sku','categoria','cantidad_vendida']].head()
```

```
['fecha', 'tienda', 'sku', 'cantidad_vendida', 'precio_venta_unitario',
```

```
'descuento_aplicado', 'venta_neta', 'mes', 'categoria']
```

[56]:

	sku	categoria	cantidad_vendida
0	HM000088	Camisetas	1
1	HM000096	Abrigos	1
2	HM000100	Accesorios	2
3	HM000088	Camisetas	6
4	HM000084	Accesorios	2

[57]:
```
inventario.columns.tolist()
inventario.head()
```

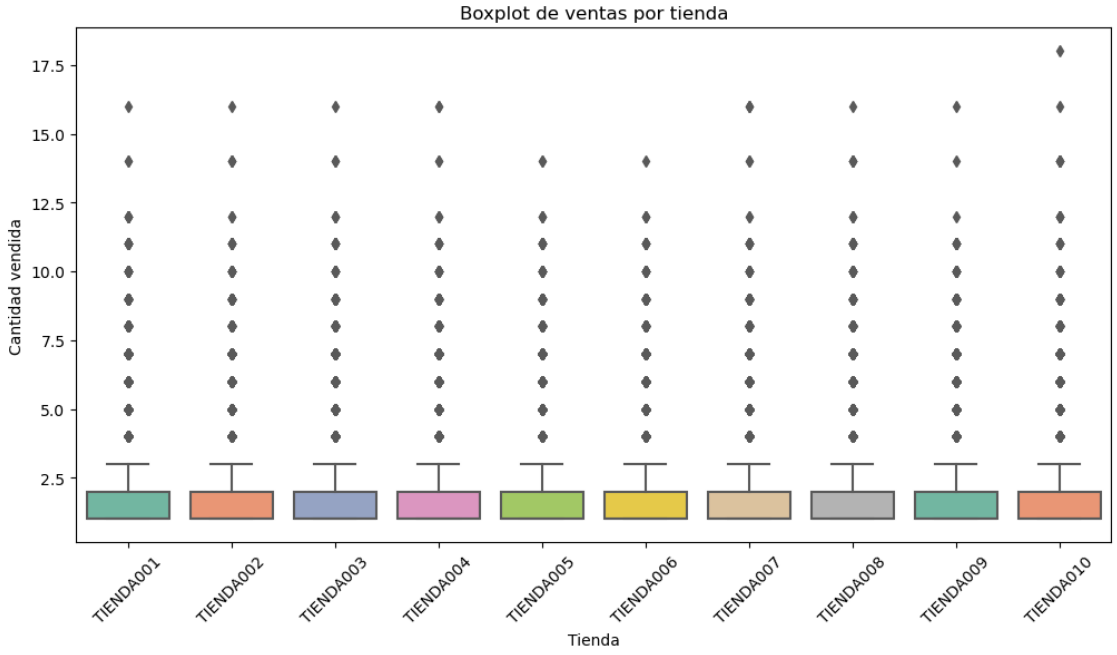
[57]:

	tienda	sku	stock_actual	stock_seguridad_actual	\
0	TIENDA001	HM000084	18	16	
1	TIENDA001	HM000054	54	23	
2	TIENDA001	HM000071	62	17	
3	TIENDA001	HM000046	7	14	
4	TIENDA001	HM000045	77	21	

	stock_en_transito	fecha_ultima_reposicion
0	16	2024-01-06
1	20	2023-12-30
2	26	2024-01-03
3	23	2024-01-05
4	12	2023-12-29

[63]:
```
ventas_cat = ventas_cat.drop(columns=['tienda_x', 'tienda_y'], errors='ignore')
```

[64]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))
sns.boxplot(data=ventas_cat, x='tienda', y='cantidad_vendida', palette='Set2')
plt.xticks(rotation=45)
plt.title("Boxplot de ventas por tienda")
plt.xlabel("Tienda")
plt.ylabel("Cantidad vendida")
plt.show()
```



4 4. Análisis temporal de ventas (2023)

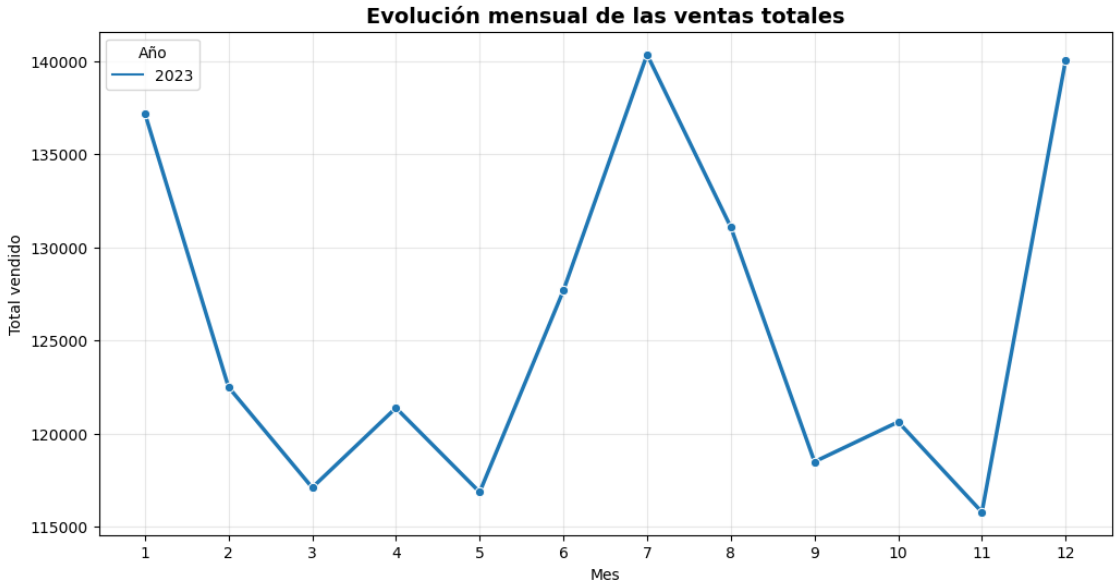
[67]:
```
# ================================================================
# 4. ANÁLISIS TEMPORAL DE VENTAS
# ================================================================

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Asegurar formato de fecha
ventas_cat['fecha'] = pd.to_datetime(ventas_cat['fecha'], errors='coerce')

# Extraer año, mes y día
ventas_cat['año'] = ventas_cat['fecha'].dt.year
ventas_cat['mes'] = ventas_cat['fecha'].dt.month
ventas_cat['dia'] = ventas_cat['fecha'].dt.day

# Agrupar ventas totales por mes
ventas_mensuales = (
    ventas_cat.groupby(['año','mes'])
    .agg(total_vendido=('cantidad_vendida','sum'))
    .reset_index()
)
```

```
# Tabla resumen con promedio y desviación
resumen_temporal = (
    ventas_mensuales.groupby('año')
    .agg(
        venta_promedio_mensual=('total_vendido','mean'),
        desviacion_mensual=('total_vendido','std'),
        max_venta=('total_vendido','max'),
        min_venta=('total_vendido','min')
    )
    .reset_index()
)

display(ventas_mensuales.head())
display(resumen_temporal)
```

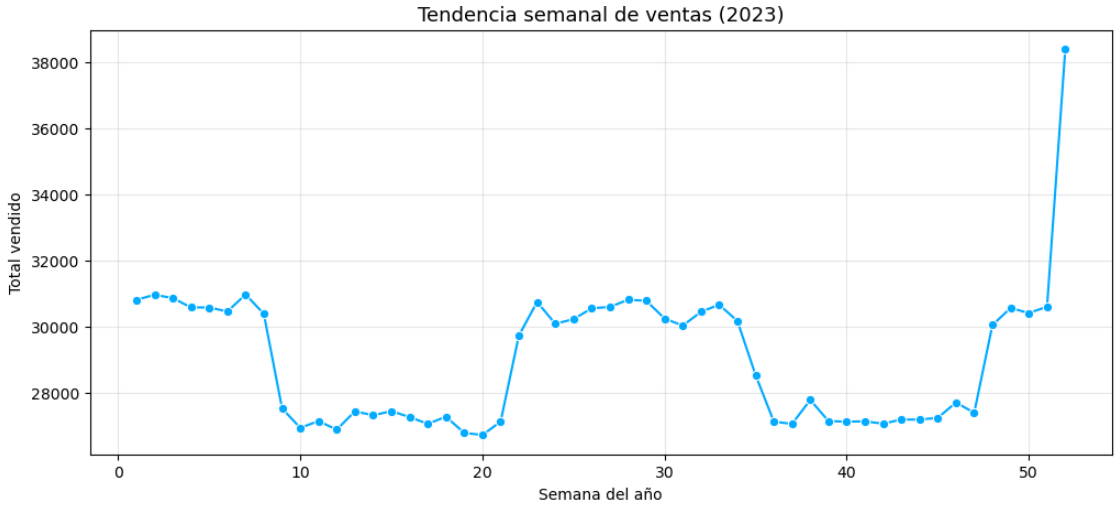
	año	mes	total_vendido
0	2023	1	137197
1	2023	2	122495
2	2023	3	117106
3	2023	4	121381
4	2023	5	116863

	año	venta_promedio_mensual	desviacion_mensual	max_venta	min_venta
0	2023	125757.666667	9252.175335	140341	115792

[68]:
```
# ================================================================
# VISUALIZACIÓN TEMPORAL
# ================================================================

plt.figure(figsize=(12,6))
sns.lineplot(
    data=ventas_mensuales,
    x='mes', y='total_vendido', hue='año',
    marker='o', linewidth=2.5, palette='tab10'
)
plt.title("Evolución mensual de las ventas totales", fontsize=14,␣
 ↪fontweight='bold')
plt.xlabel("Mes")
plt.ylabel("Total vendido")
plt.xticks(range(1,13))
plt.grid(alpha=0.3)
plt.legend(title='Año')
plt.show()
```



[71]:
```
# ================================================================
# ANÁLISIS POR SEMANA
# ================================================================

# Crear número de semana en formato entero
ventas_cat['semana'] = ventas_cat['fecha'].dt.isocalendar().week.astype(int)

# Agrupar por semana
ventas_semanal = (
    ventas_cat.groupby('semana')
    .agg(total_vendido=('cantidad_vendida','sum'))
    .reset_index()
    .sort_values('semana')
)

# Graficar
plt.figure(figsize=(12,5))
sns.lineplot(
    data=ventas_semanal,
    x='semana', y='total_vendido',
    marker='o', color='#00aaff'
)
plt.title("Tendencia semanal de ventas (2023)", fontsize=13)
plt.xlabel("Semana del año")
plt.ylabel("Total vendido")
plt.grid(alpha=0.3)
plt.show()
```



5 5. Integración con inventario y productos

[82]:
```
print(productos.columns.tolist())
productos.head()
```

```
['sku', 'categoria', 'subcategoria', 'talla', 'color', 'costo_unitario',
'precio_venta', 'proveedor_id', 'lead_time_dias', 'cantidad_minima_pedido',
'multiplico_pedido', 'temporada', 'margen']
```

[82]:

	sku	categoria	subcategoria	talla	color	costo_unitario	precio_venta	proveedor_id	lead_time_dias	cantidad_minima_pedido	multiplico_pedido	temporada	margen
0	HM000001	Vestidos	Temporada	XL	Negro	10.35	17.75	PROV001	18	50	10	Invierno	7.40
1	HM000002	Camisetas	Temporada	L	Rojo	34.60	40.98	PROV002	12	50	10	Verano	6.38
2	HM000003	Abrigos	Basico	L	Blanco	40.50	68.38	PROV003	12	100	25	Verano	27.88
3	HM000004	Camisetas	Temporada	S	Rojo	37.49	19.15	PROV004	8	100	50	Todo el año	-18.34
4	HM000005	Accesorios	Temporada	M	Rojo	35.92	56.41	PROV005	15	50	25	Todo el año	20.49

[83]:
```
print(inventario.columns.tolist())
inventario.head()
```

```
['tienda', 'sku', 'stock_actual', 'stock_seguridad_actual', 'stock_en_transito',
'fecha_ultima_reposicion']
```

[83]:

	tienda	sku	stock_actual	stock_seguridad_actual	stock_en_transito	fecha_ultima_reposicion
0	TIENDA001	HM000084	18	16	16	2024-01-06
1	TIENDA001	HM000054	54	23	20	2023-12-30
2	TIENDA001	HM000071	62	17	26	2024-01-03
3	TIENDA001	HM000046	7	14	23	2024-01-05
4	TIENDA001	HM000045	77	21	12	2023-12-29

[84]:
```
# ================================================================
# 5. INTEGRACIÓN FINAL CON INVENTARIO Y PRODUCTOS
# ================================================================

# Fusionar las dos columnas de tienda
ventas_total['tienda'] = ventas_total['tienda_y'].
 ↪combine_first(ventas_total['tienda_x'])

# Verificar el resultado
print("  Columna tienda unificada correctamente")
print(" Valores únicos en tienda:", ventas_total['tienda'].dropna().unique()[:
 ↪10])

# Eliminar duplicados si existen
if 'fecha' in ventas_total.columns and 'tienda' in ventas_total.columns:
    ventas_total = ventas_total.drop_duplicates(subset=['fecha', 'sku',␣
 ↪'tienda'])
elif 'tienda' in ventas_total.columns:
    ventas_total = ventas_total.drop_duplicates(subset=['sku', 'tienda'])

# Calcular métricas integradas
metricas_venta = (
    ventas_total.groupby(['sku', 'categoria_x', 'subcategoria', 'tienda'])
    .agg(
        demanda_promedio=('cantidad_vendida', 'mean'),
        desviacion_demanda=('cantidad_vendida', 'std'),
        ventas_totales=('cantidad_vendida', 'sum'),
```

```
        precio_promedio=('precio_venta_unitario', 'mean'),
        costo_unitario=('costo_unitario', 'mean'),
        precio_venta=('precio_venta', 'mean'),
        lead_time=('lead_time_dias', 'mean'),
        stock_actual=('stock_actual', 'mean'),
        stock_seguridad=('stock_seguridad_actual', 'mean'),
        stock_en_transito=('stock_en_transito', 'mean'),
        margen_promedio=('margen', 'mean')
    )
    .reset_index()
)

# Renombrar 'categoria_x' para dejarlo limpio
metricas_venta = metricas_venta.rename(columns={'categoria_x': 'categoria'})

# Mostrar resultado
display(metricas_venta.head(10))

# Exportar dataset final consolidado
metricas_venta.to_csv("metricas_venta_integradas.csv", index=False)
```

```
 Columna tienda unificada correctamente
Valores únicos en tienda: ['TIENDA001' 'TIENDA002' 'TIENDA003' 'TIENDA004'
'TIENDA005' 'TIENDA006']
```

	sku	categoria	subcategoria	tienda	demanda_promedio	\
0	HM000001	Vestidos	Temporada	TIENDA001	1.652055	
1	HM000001	Vestidos	Temporada	TIENDA002	1.652055	
2	HM000001	Vestidos	Temporada	TIENDA003	1.652055	
3	HM000001	Vestidos	Temporada	TIENDA004	1.652055	
4	HM000001	Vestidos	Temporada	TIENDA005	1.652055	
5	HM000001	Vestidos	Temporada	TIENDA006	1.652055	
6	HM000002	Camisetas	Temporada	TIENDA001	1.520548	
7	HM000002	Camisetas	Temporada	TIENDA002	1.520548	
8	HM000002	Camisetas	Temporada	TIENDA003	1.520548	
9	HM000002	Camisetas	Temporada	TIENDA004	1.520548	

	desviacion_demanda	ventas_totales	precio_promedio	costo_unitario	\
0	1.127590	603	37.05	10.35	
1	1.127590	603	37.05	10.35	
2	1.127590	603	37.05	10.35	
3	1.127590	603	37.05	10.35	
4	1.127590	603	37.05	10.35	
5	1.127590	603	37.05	10.35	
6	0.846934	555	91.37	34.60	
7	0.846934	555	91.37	34.60	
8	0.846934	555	91.37	34.60	
9	0.846934	555	91.37	34.60	

```
   precio_venta  lead_time  stock_actual  stock_seguridad  stock_en_transito  \
0         17.75       18.0          68.0             15.0               22.0
1         17.75       18.0          12.0             17.0               39.0
2         17.75       18.0           1.0             24.0                0.0
3         17.75       18.0          70.0             24.0                0.0
4         17.75       18.0          17.0             17.0               44.0
5         17.75       18.0          56.0             21.0               25.0
6         40.98       12.0          86.0             23.0               30.0
7         40.98       12.0          86.0             20.0               25.0
8         40.98       12.0          28.0             22.0               41.0
9         40.98       12.0          41.0             16.0                1.0

   margen_promedio
0             7.40
1             7.40
2             7.40
3             7.40
4             7.40
5             7.40
6             6.38
7             6.38
8             6.38
9             6.38
```

6 6. Análisis de los escenarios logísticos

[90]:
```python
import numpy as np
import pandas as pd

# Verificar estructura
print("Columnas disponibles en costos:\n", costos.columns.tolist())

# Parámetro para el nivel de servicio (puede adaptarse según política)
nivel_servicio_dict = {
    'Escenario 1': 0.90,
    'Escenario 2': 0.93,
    'Escenario 3': 0.95,
    'Escenario 4': 0.97,
    'Escenario 5': 0.99
}

resultados = []

for _, row in costos.iterrows():
    escenario = row['escenario']
    S = row['costo_pedido']
```

```
    h = row['costo_mantenimiento_anual']
    c_alm = row['costo_almacenamiento_m2']
    c_transf = row['costo_transferencia_tienda']
    c_rotura = row['costo_rotura_stock']
    c_obs = row['costo_obsolescencia']
    Z = 1.65  # equivalente a 95% de nivel de servicio

    print(f"\n Procesando {escenario} | S={S} | h={h}")

    df = metricas_venta.copy()

    # ================================
    # PARTE 1: Parámetros básicos
    # ================================
    df['D'] = df['demanda_promedio'] * 52  # semanal -> anual
    df['H'] = h * df['costo_unitario']

    # ================================
    # PARTE 2: Modelo (s, Q)
    # ================================
    df['Q_opt'] = np.sqrt((2 * df['D'] * S) / df['H'])
    df['D_L'] = df['demanda_promedio'] * df['lead_time']
    df['sigma_L'] = df['desviacion_demanda'] * np.sqrt(df['lead_time'])
    df['s_reorder'] = df['D_L'] + Z * df['sigma_L']

    # ================================
    # PARTE 3: Costos extendidos
    # ================================
    df['CTA_basico'] = (df['D'] / df['Q_opt']) * S + (df['Q_opt'] / 2) * df['H']
    df['CTA_almacen'] = c_alm * (df['stock_actual'] / 100)
    df['CTA_transf'] = c_transf * np.random.uniform(0.5, 1.5, len(df))  #␣
↪estimación relativa
    df['CTA_rotura'] = c_rotura * np.maximum(0, df['s_reorder'] -␣
↪df['stock_actual'])
    df['CTA_obs'] = c_obs * np.maximum(0, df['stock_actual'] - df['s_reorder'])

    # Costo total extendido
    df['CTA_total'] = df['CTA_basico'] + df['CTA_almacen'] + df['CTA_transf'] +␣
↪df['CTA_rotura'] + df['CTA_obs']

    df['escenario'] = escenario
    resultados.append(df)

# Consolidar escenarios
df_escenarios = pd.concat(resultados, ignore_index=True)

# ================================
```

```python
# PARTE 4: Resumen comparativo
# ==============================
resumen = (
    df_escenarios.groupby('escenario')
    .agg(
        Q_medio=('Q_opt', 'mean'),
        s_medio=('s_reorder', 'mean'),
        CTA_promedio=('CTA_total', 'mean'),
        CTA_total=('CTA_total', 'sum')
    )
    .reset_index()
)

display(resumen.head(10))
```

```
Columnas disponibles en costos:
 ['escenario', 'costo_pedido', 'costo_mantenimiento_anual',
'costo_almacenamiento_m2', 'costo_transferencia_tienda', 'costo_rotura_stock',
'costo_obsolescencia']

  Procesando Escenario 1 | S=10.0 | h=0.15

  Procesando Escenario 2 | S=40.0 | h=0.2

  Procesando Escenario 3 | S=50.0 | h=0.25

  Procesando Escenario 4 | S=55.0 | h=0.3

  Procesando Escenario 5 | S=60.0 | h=0.35
```

	escenario	Q_medio	s_medio	CTA_promedio	CTA_total
0	Escenario 1	21.581585	27.638714	127.560299	76536.179363
1	Escenario 2	37.380401	27.638714	246.940503	148164.301853
2	Escenario 3	37.380401	27.638714	306.341780	183805.067890
3	Escenario 4	35.789009	27.638714	348.558218	209134.930609
4	Escenario 5	34.607527	27.638714	390.123446	234074.067453

[ ]: