

# ANÁLISIS EXPLORATORIO DE DATOS DEL CONTROL DE CALIDAD EN DESPACHO DE PULPAS DE FRUTA CONCENTRADA EN TAMBORES DE 250 KG Y TOTES DE 100 KG BAJO LA NORMA CHILENA 44

## ▼ INTRODUCCIÓN

El presente trabajo desarrolla un Análisis Exploratorio de Datos (EDA) sobre registros históricos de control de calidad en el proceso de preparación y despacho de pulpas de fruta concentrada, en formatos de tambores de 250 kg y totes de 100 kg, bajo la aplicación de la Norma Chilena 44.

Los datos analizados provienen del registro operativo, el cual es completado diariamente por los analistas de calidad durante la ejecución del proceso de despacho, combinando información levantada en terreno, en tiempo real, con datos obtenidos desde el sistema SAP, tales como pedidos, packing list y observaciones de planta. Esta integración permite una trazabilidad completa del producto, desde su revisión hasta la decisión final de despacho.

El conjunto de datos contiene información relacionada con volumen de carga, muestreo según norma, defectos interiores y exteriores, acciones correctivas aplicadas y resultado final del proceso. En particular, la variable "Destino" constituye un elemento clave del análisis, ya que representa la decisión final de calidad, clasificando cada registro en categorías como producto liberado para despacho o producto destinado a decomiso, reflejando directamente el cumplimiento de los criterios sanitarios y operativos establecidos.

El objetivo de este EDA es comprender la estructura, calidad y comportamiento de los datos, identificar patrones y relaciones relevantes entre las variables, detectar posibles inconsistencias y generar una base sólida para etapas posteriores de análisis avanzado y modelamiento predictivo, orientadas a apoyar la toma de decisiones y la mejora continua del proceso de control de calidad en despacho.

## ▼ PASO 1 — IMPORTACIÓN Y CONFIGURACIÓN DEL ENTORNO

```
# -----
# PASO 1 | IMPORTACIÓN DE LIBRERÍAS Y CONFIGURACIÓN GENERAL
# -----  
  
import pandas as pd                      # Manipulación y análisis de datos  
import numpy as np                        # Operaciones numéricas  
import requests                           # Descarga de archivos desde la web  
import io                                 # Manejo de datos en memoria  
import matplotlib.pyplot as plt           # Gráficos base  
import seaborn as sns                     # Gráficos estadísticos  
from tabulate import tabulate             # Tablas formateadas en consola  
  
# -----  
# CONFIGURACIÓN VISUAL GLOBAL  
# -----  
sns.set_theme(  
    style="whitegrid",                    # Fondo claro con grilla suave  
    context="notebook"                  # Tamaños adecuados para análisis  
)  
  
plt.rcParams["figure.figsize"] = (10, 5)   # Tamaño estándar de gráficos  
plt.rcParams["axes.titlesize"] = 14         # Tamaño del título  
plt.rcParams["axes.labelsize"] = 12          # Tamaño de etiquetas
```

## ▼ PASO 2 — CARGA DE DATOS Y VALIDACIÓN INICIAL

```
# -----
# PASO 2 | CARGA DE DATOS DESDE GOOGLE SHEETS
# -----  
  
# URL directa para exportar el archivo Excel desde Google Sheets
file_url = (
    "https://docs.google.com/spreadsheets/d/"
    "1Jwrs_Ggu8WP3I866osi7szwEU3Ak4bHh/export?format=xlsx"
)  
  
# -----  
# DESCARGA DEL ARCHIVO
# -----  
response = requests.get(file_url)           # Solicitud HTTP al servidor
response.raise_for_status()                  # Verifica que la descarga fue exitosa  
  
# -----  
# LECTURA DEL EXCEL EN MEMORIA
# -----  
df = pd.read_excel(
    io.BytesIO(response.content),            # Contenido leído desde memoria
    skiprows=3,                            # Datos reales comienzan en fila 4
    engine="openpyxl"                      # Motor para archivos .xlsx
)  
  
# -----  
# VALIDACIÓN INICIAL DEL DATASET
# -----  
print("✅ Dataset cargado correctamente\n")
```

```

print("⬇ Dimensiones del dataset:")
print(f"Filas: {df.shape[0]} | Columnas: {df.shape[1]}\n")

print("⭐ Tipos de datos iniciales:")
df.info()

```

✓ Dataset cargado correctamente

⬇ Dimensiones del dataset:  
Filas: 41232 | Columnas: 28

⭐ Tipos de datos iniciales:

#	Column	Non-Null Count	Dtype
0	Fecha de la Revisión	41227	datetime64[ns]
1	Nombre de Producto	41227	object
2	Cliente	41227	object
3	Packing list	41227	object
4	Nº Pedido	41143	object
5	Peso total / Kg	41224	float64
6	Total tambores /Totes	41226	float64
7			
8	Total tambores según NormaCh44 / totes a revisar según Norma	41227	float64
9	Total tambores con defecto exterior (no revisado por norma)	41228	float64
10	Total Tambores/tote con defectos revisados por norma y defecto exterior	41224	float64
11	Tambor Greif	41227	object
12	Total tambores doble uso	33513	float64
13	Cantidad	41066	float64
14	Cod	39190	float64
15	Defecto	39190	object
16	Acción	39166	object
17	Destino	41231	object
18	Nº Tambor/ Tote	39117	object
19	Línea	39069	float64
20	CODIGO SAP	36514	object
21	Fecha Envasado	39097	object
22	Kilos	38832	float64
23	FECHA BASE FIFO	1625	object
24	PROVEEDOR TAMBOR	10662	object
25	Analista que preparación	41227	object
26	Fecha despacho	41226	datetime64[ns]
27	Analista que despacha	41226	object
	Observacion	17723	object

dtypes: datetime64[ns](2), float64(10), object(16)  
memory usage: 8.8+ MB

## ✓ PASO 3 — LIMPIEZA ESTRUCTURAL BÁSICA

```

# =====
# PASO 3 | LIMPIEZA ESTRUCTURAL BÁSICA
# =====

# -----
# ELIMINAR FILAS VACÍAS
# -----
df = df.dropna(how="all")           # Elimina filas completamente vacías

# -----
# ELIMINAR COLUMNAS VACÍAS
# -----
df = df.dropna(axis=1, how="all")    # Elimina columnas completamente vacías

# -----
# ELIMINAR DUPLICADOS
# -----
df = df.drop_duplicates()          # Elimina registros duplicados

# -----
# REINICIAR ÍNDICE
# -----
df = df.reset_index(drop=True)      # Reordena el índice desde 0

# -----
# VALIDACIÓN POST-LIMPIEZA
# -----
print("✓ Limpieza estructural completada\n")

print("⬇ Dimensiones después de la limpieza:")
print(f"Filas: {df.shape[0]} | Columnas: {df.shape[1]}\n")

print("⌚ Vista previa después de la limpieza:")
df.head()

```

Limpieza estructural completada

⚠ Dimensiones después de la limpieza:  
Filas: 41220 | Columnas: 28

🔍 Vista previa después de la limpieza:

Fecha de la Revisión	Nombre de Producto	Cliente	Packing list	Nº Pedido	Peso total / Kg	Total tambores / Totes	\nTotal tambores según NormaCh44 / totes a revisar según Norma	Total tambores con defecto exterior (no revisado por norma)	Total Tambores/tote con defectos revisados por norma y defecto exterior	... Linea	C
0 2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z		2002164244 44824096	0.0	0.0	0.0	0.0	0.0	...	5.0
1 2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z		2002164244 44824096	0.0	0.0	0.0	0.0	0.0	...	5.0
2 2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z		2002164238 44824096	0.0	0.0	0.0	0.0	0.0	...	2.0
3 2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z		2002164236 44824096	19122.0	80.0	13.0	0.0	0.0	...	Nan
4 2024-09-25	PASTA TOMATE HB 30/32 NTSS 1,0 MM 55GAL	RPB S.A.	2002168387	46044675	25076.0	104.0	104.0	19.0	27.0	...	8.0

5 rows × 28 columns

## ▼ PASO 4 — LIMPIEZA SEMÁNTICA DEL DATASET

```
# =====#
# PASO 4 | LIMPIEZA SEMÁNTICA DEL DATASET (VERSIÓN FINAL)
# =====#

def limpiar_dataset(df):
    """
    Limpia y estandariza el DataFrame:
    - Normaliza nombres de columnas
    - Convierte fechas y numéricos
    - Elimina columnas irrelevantes
    - Imputa valores lógicos
    - Convierte 'destino' a binaria
    - Deja 'destino' como última columna
    """
    df = df.copy() # Crea una copia para no modificar el dataset original

    # -----
    # LIMPIEZA DE NOMBRES DE COLUMNAS
    # -----
    df.columns = (
        df.columns
        .str.strip() # Elimina espacios al inicio y final
        .str.lower() # Convierte nombres a minúsculas
        .str.replace(r'^\w\s', '', regex=True) # Elimina símbolos especiales
        .str.replace(' ', '_') # Reemplaza espacios por guiones bajos
    )

    # -----
    # CONVERSIÓN DE COLUMNAS FECHA
    # -----
    columnas_fecha = [ # Lista de columnas que deberían ser fechas
        'fecha_de_la_revisin',
        'fecha_envasado',
        'fecha_despacho'
    ]

    for col in columnas_fecha:
        if col in df.columns: # Recorre cada columna de fecha
            if col in df.columns: # Verifica que la columna exista
                df[col] = pd.to_datetime(df[col], errors='coerce') # Convierte a fecha, errores → NaT

    # -----
    # CONVERSIÓN DE COLUMNAS NUMÉRICAS
    # -----
    columnas_numericas = df.select_dtypes(include=['int64', 'float64']).columns # Detecta columnas numéricas
    df[columnas_numericas] = df[columnas_numericas].apply(
```

```

        pd.to_numeric, errors='coerce'                                # Fuerza valores numéricos, errores → NaN
    )

# -----
# IMPUTACIÓN LÓGICA DE VALORES FALTANTES
# -----
if 'total_tambores_doble_uso' in df.columns:                      # Verifica si la columna existe
    df['total_tambores_doble_uso'] = (                               # Asume 0 cuando no hay información
        df['total_tambores_doble_uso']
        .fillna(0)                                                 # Convierte a entero
        .astype(int)
    )

# -----
# CONVERSIÓN DE 'DESTINO' A BINARIA
# -----
if 'destino' in df.columns:                                       # Verifica existencia de la columna
    # Reemplazo categórico → binario
    df['destino'] = df['destino'].replace({                         # Mapea texto a valores binarios
        'Decomiso': 1,
        'Liberado para Despacho': 0
    })

    # Conversión a entero nullable (permite NaN)
    df['destino'] = df['destino'].astype(pd.Int64Dtype()) # Permite enteros con valores faltantes

# -----
# ELIMINACIÓN DE COLUMNAS NO RELEVANTES
# -----
columnas_eliminar = [                                           # Columnas que no aportan al análisis
    'fecha_base_fifo',
    'proveedor_tambor',
    'observacion',
    'codigo_sap'
]

df = df.drop(columns=[c for c in columnas_eliminar if c in df.columns]) # Elimina solo si existen

# -----
# MOVER 'DESTINO' AL FINAL
# -----
if 'destino' in df.columns:                                       # Verifica que existe 'destino'
    columnas = [c for c in df.columns if c != 'destino'] # Todas menos 'destino'
    columnas.append('destino')                            # Agrega 'destino' al final
    df = df[columnas]                                    # Reordena columnas

return df                                                       # Retorna el DataFrame limpio

# -----
# REAPLICAR LIMPIEZA SEMÁNTICA
# -----
df = limpiar_dataset(df)                                         # Aplica la función al dataset

# -----
# VALIDACIÓN DEL AJUSTE
# -----
print("✅ Limpieza semántica aplicada correctamente\n") # Mensaje de confirmación

print("Valores únicos en 'total_tambores_doble_uso':") # Título descriptivo
print(df['total_tambores_doble_uso'].value_counts().head()) # Muestra frecuencias principales

print("\nValores en la variable objetivo 'destino':") # Título descriptivo
print(df['destino'].value_counts(dropna=False)) # Conteo incluyendo valores nulos

```

Limpieza semántica aplicada correctamente

```

Valores únicos en 'total_tambores_doble_uso':
total_tambores_doble_uso
0      41168
80      33
28      3
20      3
1       3
Name: count, dtype: int64

Valores en la variable objetivo 'destino':
destino
0      38825
1      2394
<NA>      1
Name: count, dtype: Int64
/tmp/ipython-input-2885632191.py:65: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a fut
    df['destino'] = df['destino'].replace({                           # Mapea texto a valores binarios

```

## ▼ PASO 5 — ANÁLISIS DE CALIDAD DE DATOS (VALORES FALTANTES)

```

# =====
# PASO 5 | ANÁLISIS DE CALIDAD DE DATOS
# =====

# -----
# CÁLCULO DE VALORES FALTANTES
# -----
tabla_faltantes = (
    pd.DataFrame({
        "valores_faltantes": df.isna().sum(),                         # Conteo de NA

```

```

        "porcentaje_faltantes": (df.isna().mean() * 100) # Porcentaje de NA
    })
    .query("valores_faltantes > 0") # Solo columnas con NA
    .sort_values("porcentaje_faltantes", ascending=False) # Orden descendente
    .round(2) # Redondeo
)

# -----
# MOSTRAR TABLA ÚNICA
# -----
print("📊 Tabla de valores faltantes:\n")
tabla_faltantes

```

📊 Tabla de valores faltantes:

	valores_faltantes	porcentaje_faltantes	grid icon	edit icon
kilos	2392	5.80		
linea	2153	5.22		
fecha_ensvasado	2127	5.16		
n_tambor_tote	2108	5.11		
acción	2059	5.00		
cod	2035	4.94		
defecto	2035	4.94		
cantidad	163	0.40		
n_pedido	86	0.21		
total_tamborestote_con_defectos_revisados_por_norma_y_defecto_exterior	5	0.01		
peso_total_kg	5	0.01		
total_tambores_totes	3	0.01		
fecha_despacho	3	0.01		
analista_que_despacha	3	0.01		
cliente	2	0.00		
packing_list	2	0.00		
fecha_de_la_revisión	2	0.00		
nombre_de_producto	2	0.00		
total_tambores_según_normach44_totes_a_revisar_según_norma	2	0.00		
analista_que_preparación	2	0.00		
tambor_greif	2	0.00		
total_tambores_con_defecto_exterior_no_revisado_por_norma	1	0.00		
destino	1	0.00		

Pasos siguientes: [Generar código con tabla\\_faltantes](#) [New interactive sheet](#)

## PASO 6 — ANÁLISIS DESCRIPTIVO DE VARIABLES NUMÉRICAS y ESTADÍSTICA INFERENCIAL

### ▼ PASO 6.1 | ANÁLISIS DESCRIPTIVO DE VARIABLES NUMÉRICAS CLAVE

```

# =====
# PASO 6.1 | ANÁLISIS DESCRIPTIVO DE VARIABLES NUMÉRICAS CLAVE
# =====

# -----
# DEFINICIÓN DE VARIABLES NUMÉRICAS CLAVE
# -----
# Representan cantidades, pesos y conteos operativos
numeric_key_columns = [
    'peso_total_kg',
    'kilos',
    'cantidad',
    'total_tambores_totes',
    'total_tambores_según_normach44_totes_a_revisar_según_norma',
    'total_tambores_con_defecto_exterior_no_revisado_por_norma',
    'total_tamborestote_con_defectos_revisados_por_norma_y_defecto_exterior',
    "destino"
]

# -----
# ESTADÍSTICAS DESCRIPTIVAS
# -----
print("📊 Estadísticas descriptivas para variables numéricas clave:\n")

df[numeric_key_columns].describe() # Resumen estadístico completo

```

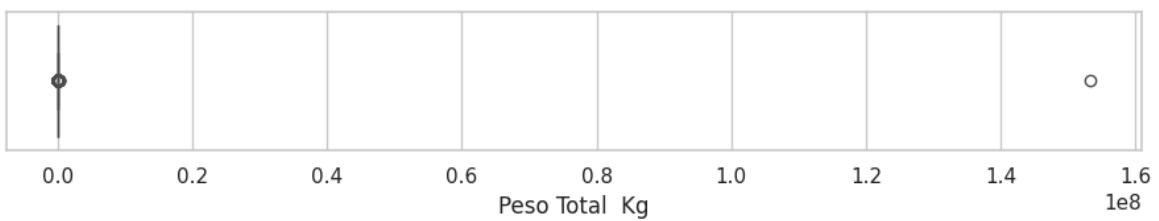
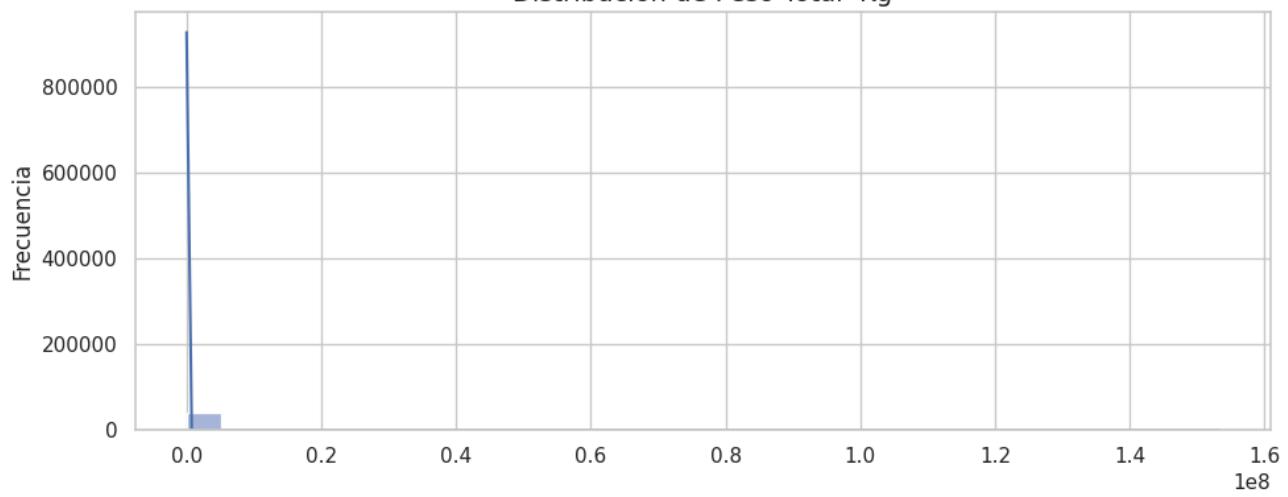
Estadísticas descriptivas para variables numéricas clave:

	peso_total_kg	kilos	cantidad	total_tambores_totes	total_tambores_según_normach44_totes_a_revisar_según_nc	
<b>count</b>	4.121500e+04	38828.000000	41057.000000	41217.000000		41218.000000
<b>mean</b>	7.439053e+03	295.001700	2.084736	29.020501		9.446000
<b>std</b>	7.552947e+05	778.242869	211.232580	2947.918660		959.169000
<b>min</b>	0.000000e+00	0.000000	0.000000	0.000000		0.000000
<b>25%</b>	0.000000e+00	235.000000	1.000000	0.000000		0.000000
<b>50%</b>	0.000000e+00	238.000000	1.000000	0.000000		0.000000
<b>75%</b>	0.000000e+00	240.000000	1.000000	0.000000		0.000000

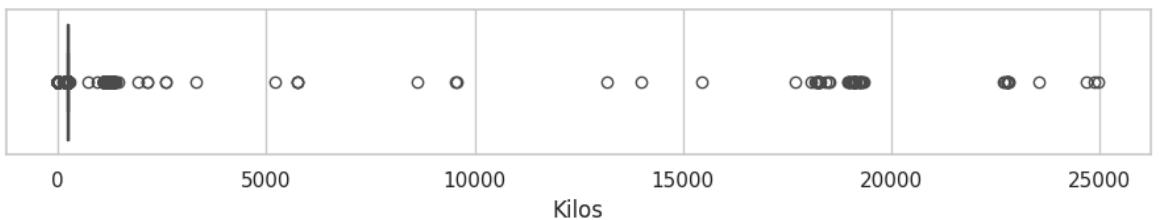
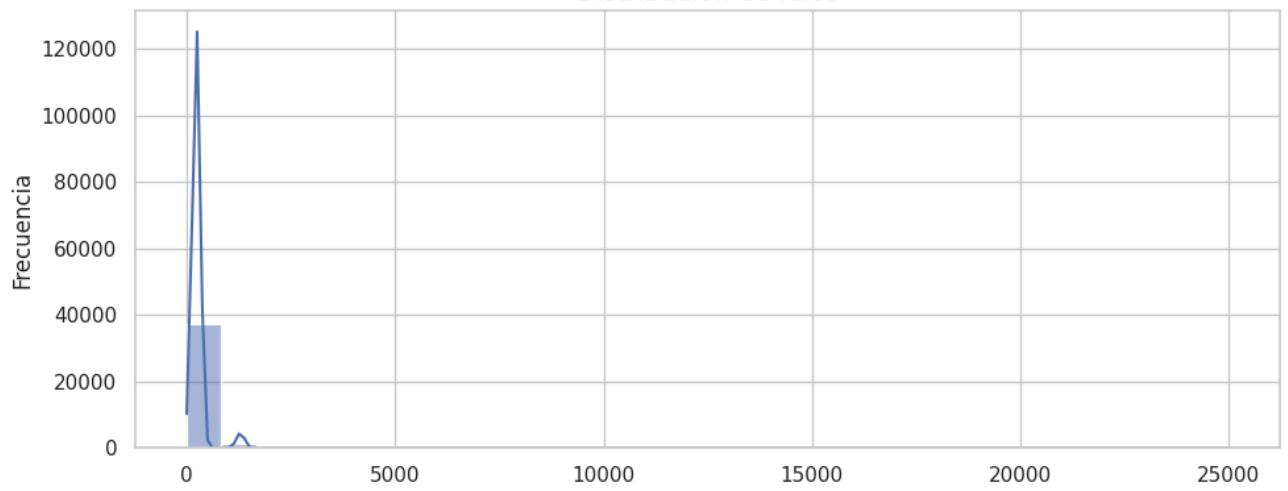
## PASO 6.2 | VISUALIZACIÓN DE VARIABLES NUMÉRICAS CLAVE



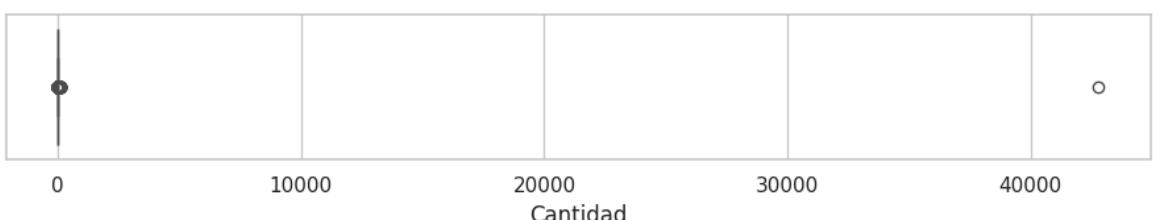
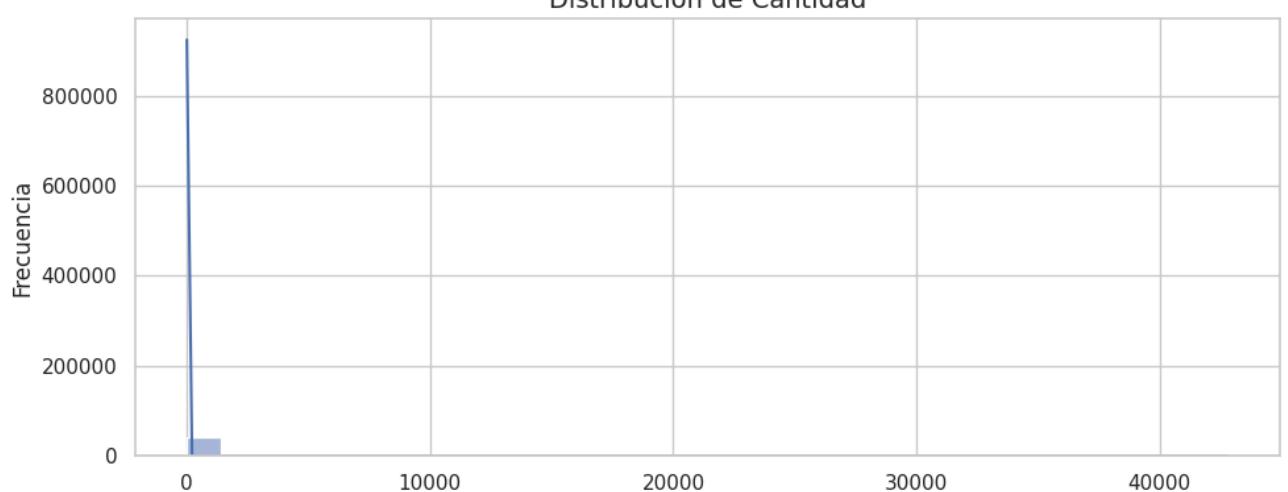
Distribución de Peso Total Kg



Distribución de Kilos

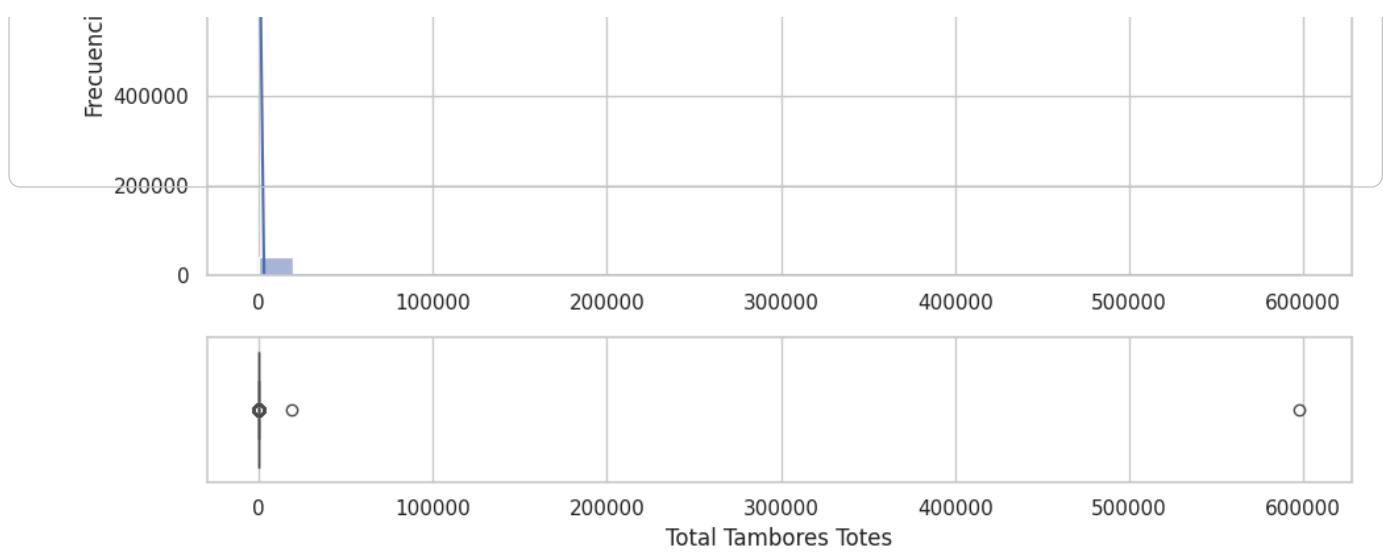


Distribución de Cantidad

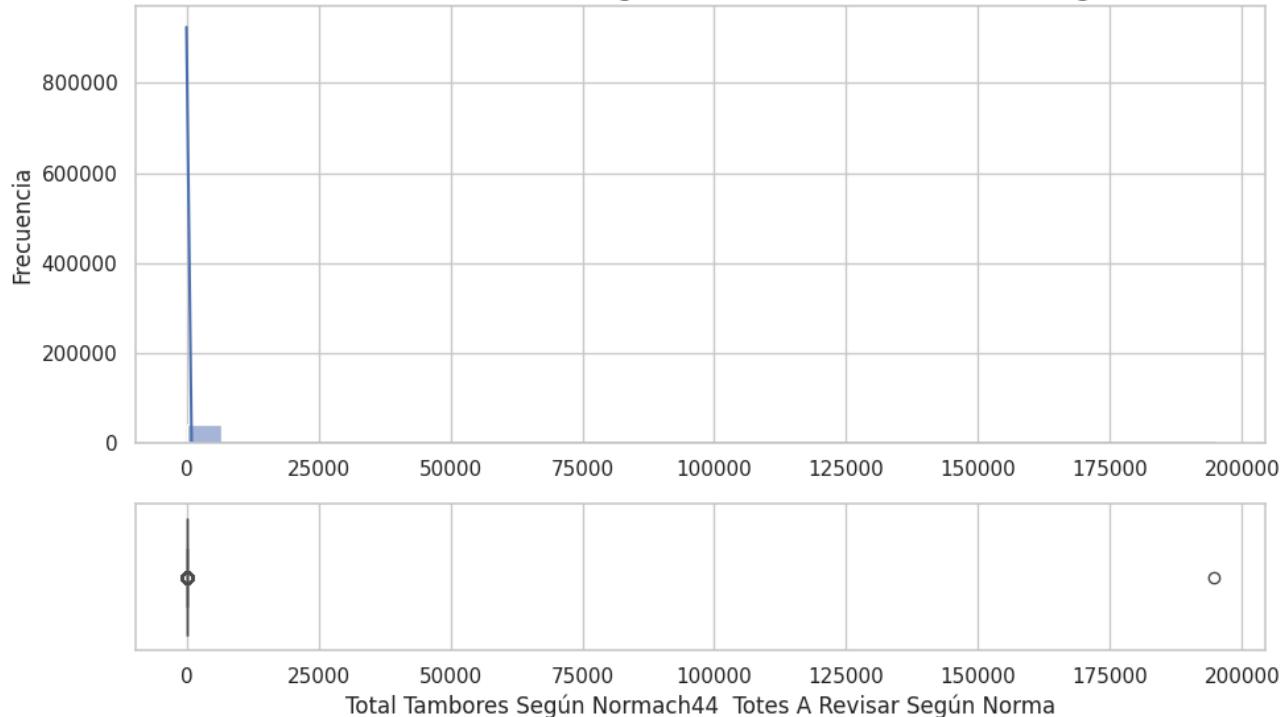


Distribución de Total Tambores Totes

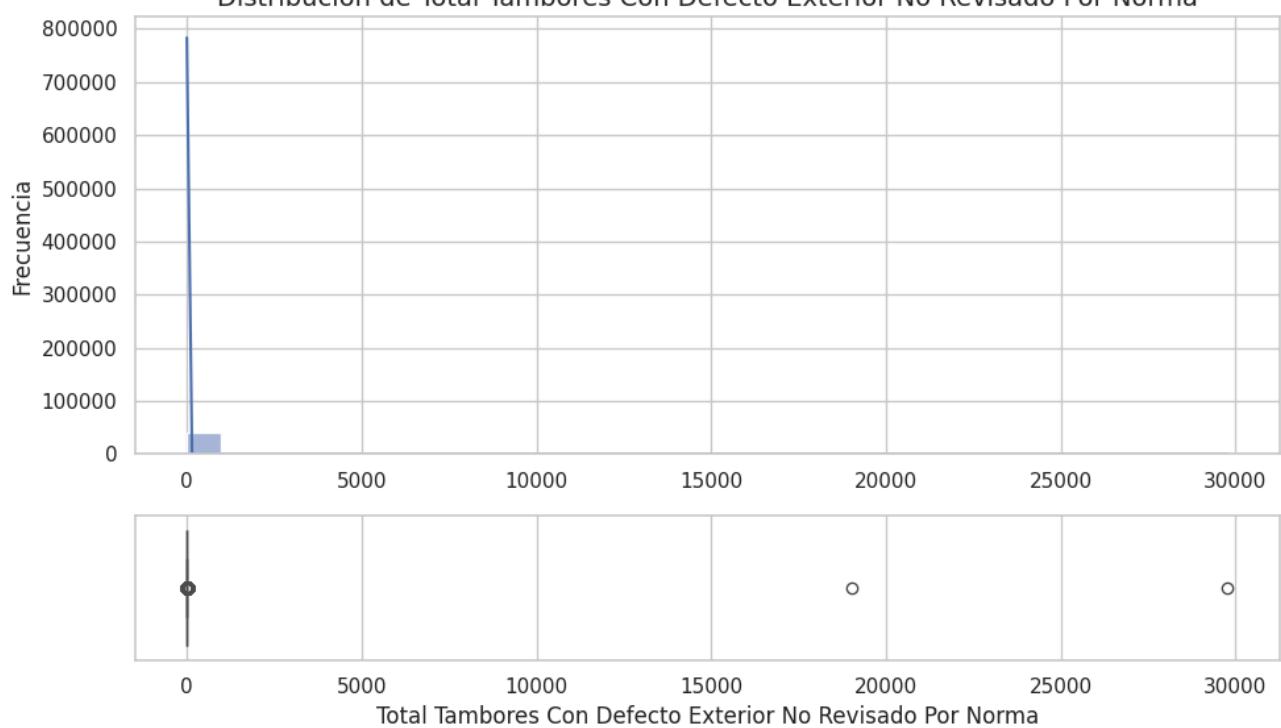




Distribución de Total Tambores Según Normach44 Totes A Revisar Según Norma



Distribución de Total Tambores Con Defecto Exterior No Revisado Por Norma



Distribución de Total Tamborestote Con Defectos Revisados Por Norma Y Defecto Exterior



## Resumen del Análisis Univariado de Variables Numéricas Clave

### 1. peso\_total\_kg

La mayoría de los registros presenta un valor de **0 kg**, lo que se refleja en una mediana y percentiles bajos iguales a cero. Sin embargo, existen **valores extremadamente altos** que generan una **media y desviación estándar muy elevadas**, indicando una distribución **fuertemente sesgada a la derecha** con outliers de gran magnitud.

### 2. kilos

#### Total Tamborestote Con Defectos Revisados Por Norma Y Defecto Exterior

Los datos se concentran principalmente en torno a los **235–240 kg**, con una dispersión moderada alrededor de la media, lo que evidencia un **sesgo positivo moderado**. Aunque el comportamiento general es más estable que otras variables, se observan **outliers relevantes** que amplían considerablemente el rango de valores.

### 3. cantidad

La **cantidad más frecuente es 1**, lo que sugiere registros mayoritariamente unitarios. No obstante, la presencia de **valores extremos muy altos** incrementa la media y la dispersión, generando una distribución **altamente asimétrica hacia la derecha**.

### 4. total\_tambores\_totes

Predominan los valores **0**, indicando que la mayoría de los registros no incluye tambores o totes. La media se ve afectada por **casos excepcionales con volúmenes muy elevados**, lo que produce una alta variabilidad y un marcado sesgo a la derecha.

### 5. total\_tambores\_segn\_normach44\_totes\_a\_revisar\_segn\_norma

Casi todos los registros presentan **0 tambores/totes a revisar**, mientras que un número reducido de observaciones con valores muy altos genera una **distribución extremadamente sesgada y con outliers significativos**.

### 6. total\_tambores\_con\_defecto\_exterior\_no\_revisado\_por\_norma

La mayoría de los datos indica **ausencia de defectos exteriores**, concentrándose en 0. Sin embargo, existen **eventos puntuales con altos conteos**, lo que incrementa la dispersión y evidencia una fuerte simetría positiva.

### 7. total\_tambores\_tote\_con\_defectos\_revisados\_por\_norma\_y\_defecto\_exterior

Esta variable también **presenta una alta concentración en 0**, reflejando pocos casos con defectos. Los **valores extremos detectados** explican la diferencia entre media y mediana y confirmar un patrón de **sesgo a la derecha**.

## Conclusión General

Todas las variables numéricas analizadas comparten una estructura de datos caracterizada por **muchos ceros y pocos valores extremos muy grandes**, lo que produce **distribuciones asimétricas y alta variabilidad**. Antes de realizar análisis inferenciales o modelamiento, es fundamental **validar los outliers**, evaluar su naturaleza (eventos reales vs. errores de registro) y considerar el uso de **medidas robustas o transformaciones** para un análisis más representativo.

## ▼ PASO 6.3 | ESTADÍSTICA INFERENCIAL COMPLETA – TABLA ÚNICA

```
# =====
# PASO 6.3 | ESTADÍSTICA INFERENCIAL · COMPLETA · · TABLA ÚNICA
# =====
import numpy as np # Cálculos numéricos
import pandas as pd # Manejo de DataFrames
from scipy.stats import shapiro, t, mannwhitneyu # Tests estadísticos

confidence_level = 0.95 # Nivel de confianza 95%
inferential_results = [] # Lista para almacenar resultados finales
# -----
# ANÁLISIS INFERENCIAL VARIABLE POR VARIABLE
# -----
for col in numeric_key_columns: # Iterar variables numéricas clave

    if col != 'destino': # Excluir variable objetivo
        # -----
        # 1. LIMPIEZA DE DATOS
        # -----
        data = df[col].dropna() # Eliminar valores faltantes
        n = len(data) # Tamaño muestral
        mean = data.mean() # Media muestral
        std = data.std() # Desviación estándar muestral
        # -----
        # 2. NORMALIDAD (SHAPIRO-WILK)
        # -----
        # Evalúa si la variable sigue una distribución normal
        shapiro_stat, shapiro_p = shapiro(data)
        # -----
        # 3. INTERVALO DE CONFIANZA 95%
        # -----
        # Se usa t-Student (varianza poblacional desconocida)
        t_crit = t.ppf((1 + confidence_level) / 2, df=n - 1)
        margin_error = t_crit * (std / np.sqrt(n))

        ic_inf = mean - margin_error # Límite inferior IC
        ic_sup = mean + margin_error # Límite superior IC
        # -----
        # 4. COMPARACIÓN SEGÚN DESTINO
        # -----
        # Test no paramétrico (robusto ante no normalidad)
        grupo_0 = df[df['destino'] == 0][col].dropna() # Destino = 0
        grupo_1 = df[df['destino'] == 1][col].dropna() # Destino = 1
        mw_stat, mw_p = mannwhitneyu(
```

	Variable	N	Media	IC 95% Inferior	IC 95% Superior	Shapiro p-value	Normalidad	p-value (Destino)	s
0	peso_total_kg	41215	7439.053185	147.003766	14731.102603	7.200906e-158	No	1.664534e-04	
1	kilos	38828	295.001700	287.260571	302.742828	7.147930e-154	No	1.261896e-29	
2	cantidad	41057	2.084736	0.041454	4.128017	9.232682e-158	No	7.295589e-27	
3	total_tambores_totes	41217	29.020501	0.560288	57.480715	7.193718e-158	No	1.888600e-06	
4	total_tambores_segun_normach11_totes_a_revisar	41218	0.116065	0.186925	18.707005	7.223606e-158	No	3.419386e-05	

Pasos siguientes: [Generar código con inferential\\_df](#) [New interactive sheet](#)

## Interpretación de la Tabla de Estadística Inferencial

Esta tabla detalla las características clave de las variables numéricas y su relación con el destino (liberado/decomisado):

- Normalidad (Shapiro p-value):** Ninguna de las variables numéricas clave sigue una distribución normal (todos los p-values son muy bajos). Esto confirma que sus distribuciones son sesgadas, como se observó en los histogramas.
- Intervalo de Confianza (IC 95%):** Los amplios intervalos de confianza en algunas variables (`peso_total_kg`, `total_tambores_totes`) reflejan la alta variabilidad y la presencia de valores extremos (outliers).
- Diferencia Significativa por Destino (p-value Mann-Whitney U):** Para la mayoría de las variables, el p-value es menor a 0.05, indicando una diferencia estadísticamente significativa en la distribución de la variable entre productos liberados y decomisados. Esto significa que los productos decomisados tienden a tener valores distintos en estas métricas en comparación con los liberados, haciendo que estas variables sean importantes para predecir el destino.

**Conclusión:** Aunque las variables numéricas no son normales, son predictoras importantes del destino del producto, ya que sus distribuciones difieren significativamente entre los ítems liberados y los decomisados (con una excepción). Esto justifica su consideración en análisis y modelos futuros.

## ✓ PASO 7 — ANÁLISIS DE VARIABLES CATEGÓRICAS

```
# =====#
# PASO 7 | ANÁLISIS DE VARIABLES CATEGÓRICAS
# =====#
# -----
# DEFINICIÓN DE VARIABLES CATEGÓRICAS CLAVE
# -----
categorical_key_columns = [
    'nombre_de_producto',
    'cliente',
    'defecto',
    'accin',
    'destino',
    'linea',
    'analista_que_preparacion',
```

```

'analista_que_despacha'

]

# Mantener solo columnas existentes en el DataFrame
categorical_key_columns = [
    col for col in categorical_key_columns if col in df.columns
]

# -----
# FRECUENCIAS ABSOLUTAS Y RELATIVAS
# -----
for col in categorical_key_columns:

    print(f"\n📊 Variable categórica: {col.replace('_', ' ').title()}\n")

    # Frecuencia absoluta
    freq_abs = df[col].value_counts(dropna=False)
    print("Frecuencia absoluta:")
    print(freq_abs)

    # Frecuencia relativa (%)
    freq_rel = df[col].value_counts(dropna=False, normalize=True) * 100
    print("\nFrecuencia relativa (%):")
    print(freq_rel.round(2))

```

#### 📊 Variable categórica: Nombre De Producto

Frecuencia absoluta:

nombre_de_producto	
PASTA TOMATE HB 30/32 BRIX ST 55GAL	6553
PASTA TOMATE HB 30/32 NTSS 55GAL 1,5MM	2563
PULPA PERA 30/32 BRIX ST 55GAL	1578
PULPA MANZANA 30/32 BRIX A/A ST 55GAL	1535
PASTA TOMATE CB 28/30 NTSS ESP 55GAL	1525
...	
PULPA DURAZNO 30/32 BX STF 55GAL	1
PULP PERA 30/32 BRIX ST 55GAL	1
PULPA DURAZNO 30/32 BRIX ST 55GAL	1
PULPA DURAZNO 30/32 BRIX ESP C 55GL	1
PULPA DURAZNO 30/32 BX CILIND ESP 55GAL	1

Name: count, Length: 239, dtype: int64

Frecuencia relativa (%):

nombre_de_producto	
PASTA TOMATE HB 30/32 BRIX ST 55GAL	15.90
PASTA TOMATE HB 30/32 NTSS 55GAL 1,5MM	6.22
PULPA PERA 30/32 BRIX ST 55GAL	3.83
PULPA MANZANA 30/32 BRIX A/A ST 55GAL	3.72
PASTA TOMATE CB 28/30 NTSS ESP 55GAL	3.70
...	
PULPA DURAZNO 30/32 BX STF 55GAL	0.00
PULP PERA 30/32 BRIX ST 55GAL	0.00
PULPA DURAZNO 30/32 BRIX ST 55GAL	0.00
PULPA DURAZNO 30/32 BRIX ESP C 55GL	0.00
PULPA DURAZNO 30/32 BX CILIND ESP 55GAL	0.00

Name: proportion, Length: 239, dtype: float64

#### 📊 Variable categórica: Cliente

Frecuencia absoluta:

cliente	
HOSH INTERNATIONAL LLC	3247
RPB S.A.	3004
KAGOME	2740
ASI SRL	2230
CBC INTL. S.A.	1120
...	
KBD SPA	1
SUN PACIFIC PRODUCTS	1
VINAGRES FINOS LTDA	1
UD CHEMIE TALINN CU	1
IND.COM.PROD.ALIMENTICIOS CEPERA	1

Name: count, Length: 358, dtype: int64

Frecuencia relativa (%):

cliente	
HOSH INTERNATIONAL LLC	7.88
RPB S.A.	7.29
KAGOME	6.65
ASI SRL	5.41
CBC INTL. S.A.	2.72
...	

## ▼ PASO 7.1—VISUALIZACIÓN — GRÁFICOS CLAROS Y COMPARABLES

```

# =====#
# VISUALIZACIÓN DE VARIABLES CATEGÓRICAS
# =====#

for col in categorical_key_columns:                      # Recorre cada columna categórica seleccionada

    # Contar categorías
    freq = df[col].value_counts(dropna=False)          # Cuenta frecuencia de cada categoría (incluye NaN)

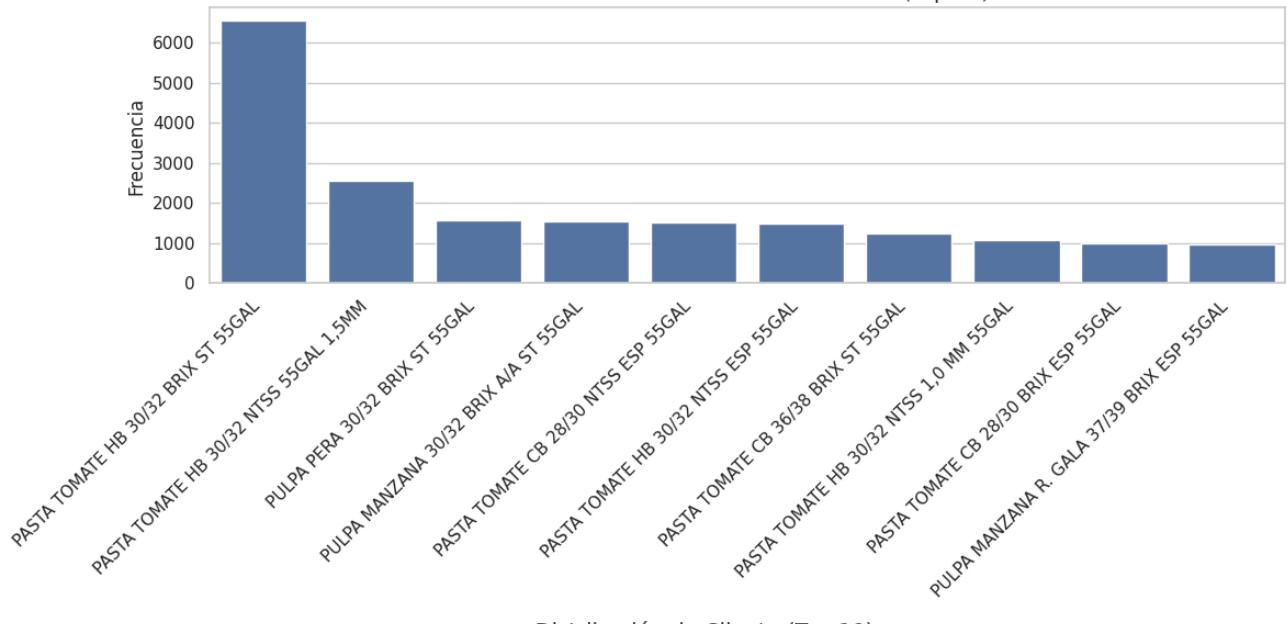
    # Seleccionar Top 10 si hay muchas categorías
    if freq.shape[0] > 10:                             # Verifica si hay más de 10 categorías
        freq = freq.head(10)                            # Mantiene solo las 10 más frecuentes
        title_suffix = " (Top 10)"                     # Agrega texto aclaratorio al título
    else:
        title_suffix = ""                                # No agrega sufijo si hay pocas categorías

```

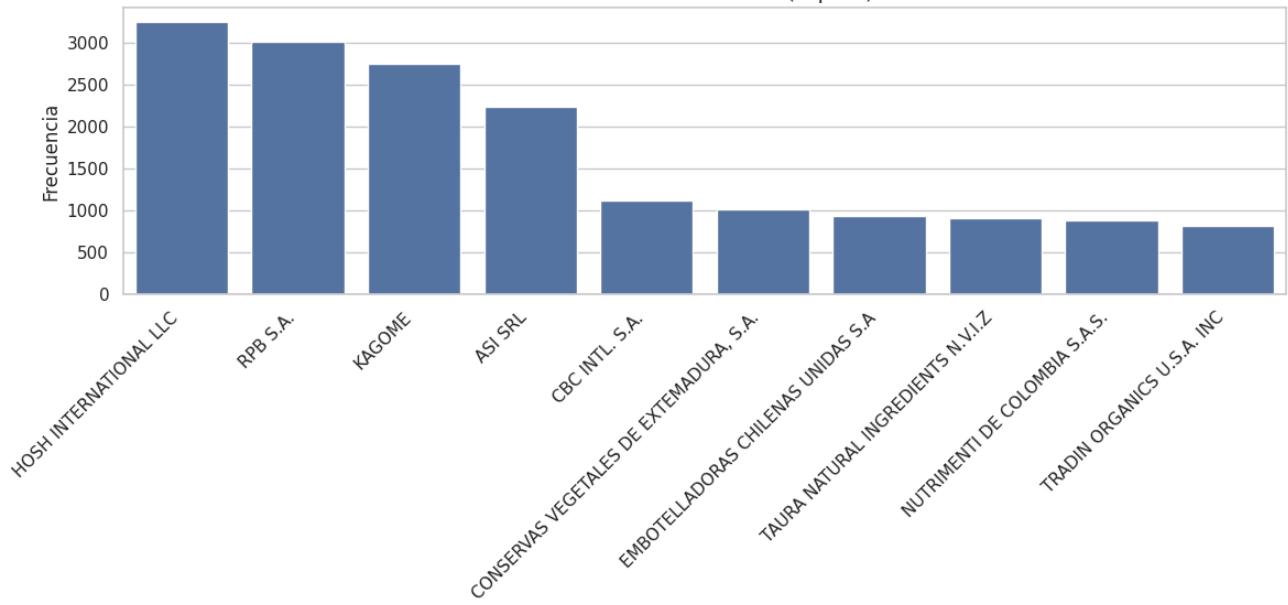




Distribución de Nombre De Producto (Top 10)

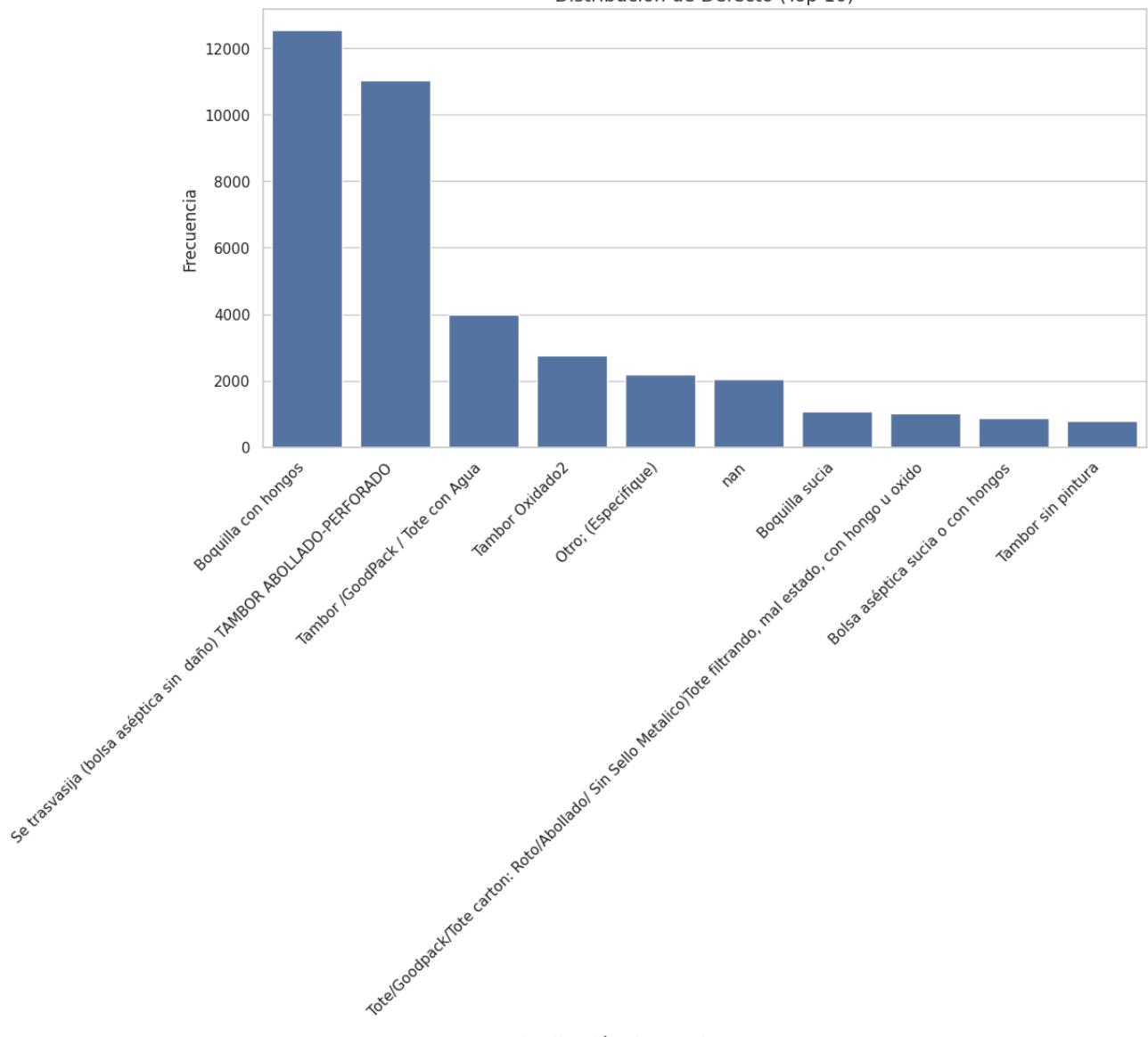


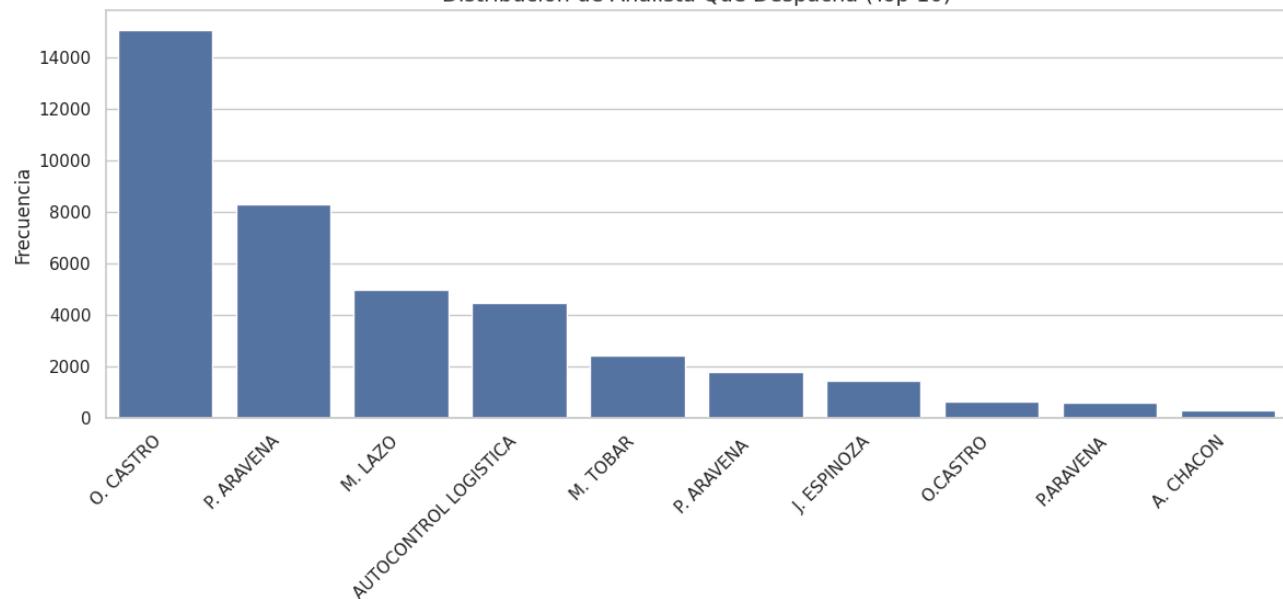
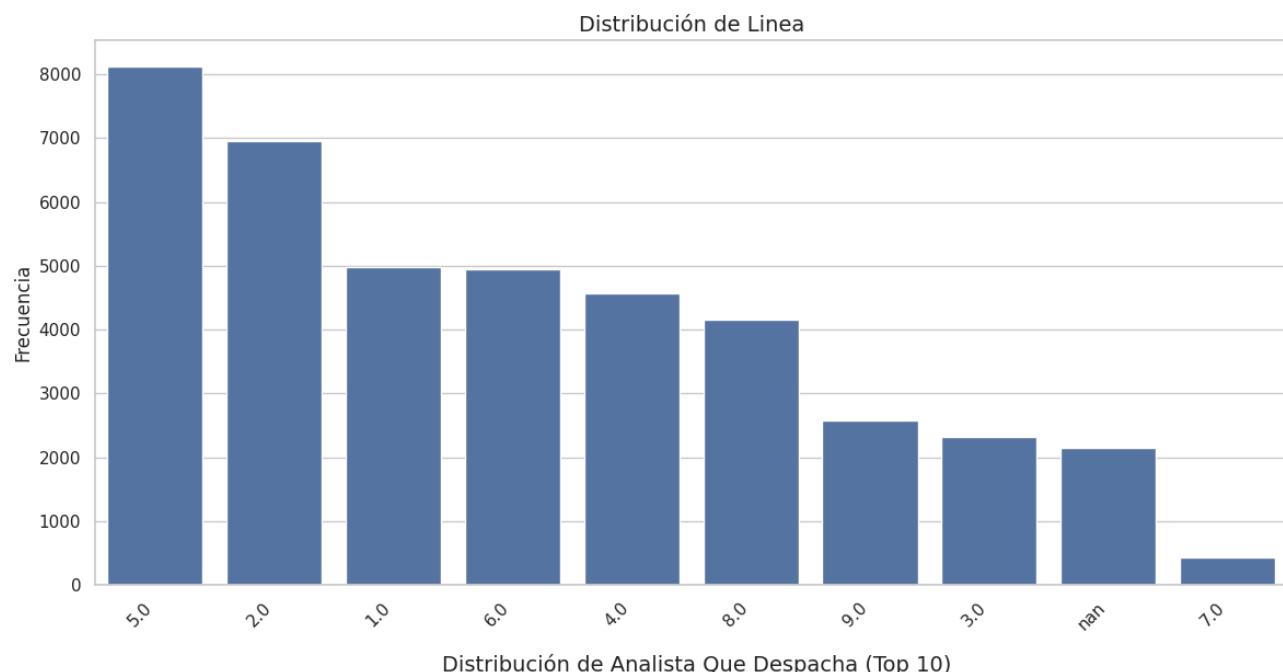
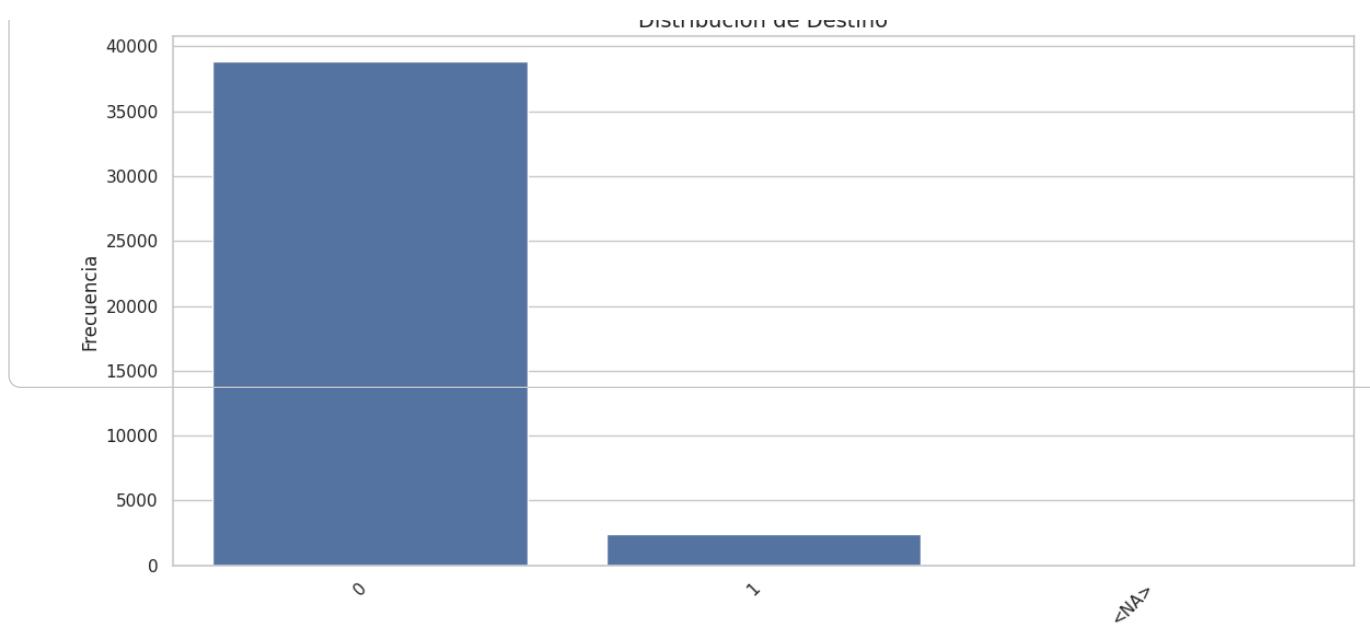
Distribución de Cliente (Top 10)



```
/tmp/ipython-input-3705730577.py:30: UserWarning: Tight layout not applied. The bottom and top margins cannot be made large enough to contain the subplots.
# Ajusta automáticamente los márgenes
```

Distribución de Defecto (Top 10)





## Interpretación de Variables Categóricas

### 1. nombre\_de\_producto

Existe una **alta diversidad de productos**, pero la actividad se concentra en unos pocos. Un número reducido de ítems domina el volumen total, destacando productos líderes, mientras que la mayoría tiene una participación marginal.

### 2. cliente

Aunque hay muchos clientes distintos, **la mayor parte del negocio proviene de un grupo reducido**. Esto refleja una estructura tipo Pareto, donde algunos clientes son estratégicamente clave para la operación.

### 3. defecto

Los defectos se concentran principalmente en **dos categorías dominantes**, lo que permite identificar claramente las **principales fuentes de problemas de calidad**. La presencia de registros sin especificación indica oportunidades de mejora en el registro de información.

### 4. destino

La gran mayoría de los productos está **liberada para despacho**, mientras que una proporción menor corresponde a **decomisos**. Este indicador es clave para evaluar la calidad global y las pérdidas asociadas.

### 5. linea

Algunas líneas de producción concentran más actividad que otras. La existencia de **valores sin línea asignada** sugiere la necesidad de mejorar la trazabilidad del proceso productivo.

### 6. analista\_que\_despacha

La actividad está concentrada en pocos analistas, lo que puede reflejar **especialización o desequilibrios de carga laboral**. Se detectan inconsistencias en la escritura de nombres, evidenciando la necesidad de **estandarización de datos**.

## Conclusión General

Las variables categóricas analizadas presentan **patrones de alta concentración**, donde pocos elementos explican la mayor parte de los registros. Este comportamiento es clave para **priorizar acciones**, mejorar el control de calidad, optimizar procesos productivos y fortalecer la gestión de clientes y recursos humanos.

## ▼ PASO 8 — ANÁLISIS TEMPORAL

### ▼ PASO 8.1 — PREPARACIÓN DE VARIABLES TEMPORALES

```
# =====#
# PASO 8.1 | PREPARACIÓN DE VARIABLES TEMPORALES
# =====#

# Asegurar que las columnas de fecha sean datetime
df['fecha_de_la_revisión'] = pd.to_datetime(df['fecha_de_la_revisión'], errors='coerce') # Fecha de revisión
df['fecha_envasado'] = pd.to_datetime(df['fecha_envasado'], errors='coerce') # Fecha de envasado
df['fecha_despacho'] = pd.to_datetime(df['fecha_despacho'], errors='coerce') # Fecha de despacho

print("✅ Columnas de fecha verificadas y preparadas")
✅ Columnas de fecha verificadas y preparadas
```

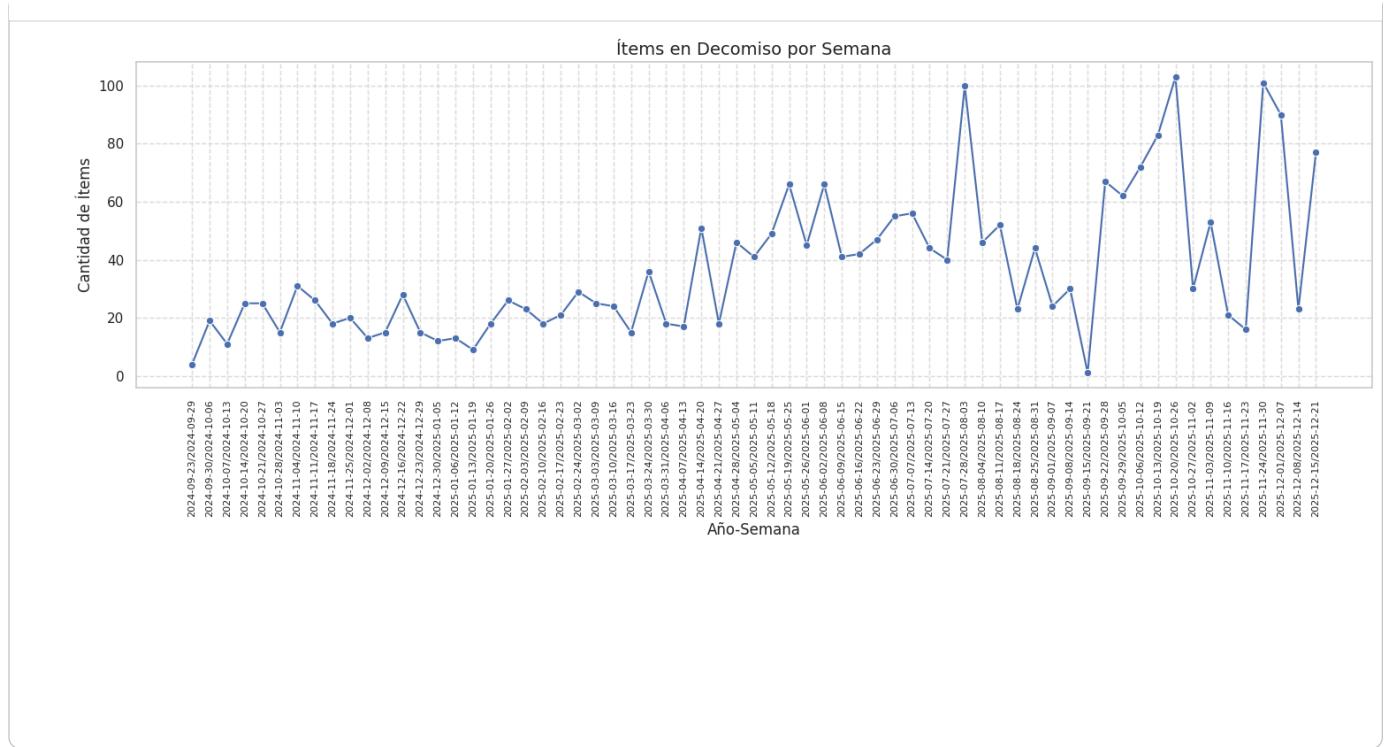
### ▼ PASO 8.2.1 | EVENTOS CRÍTICOS POR SEMANA (SECUENCIA CONTINUA)

```
# =====#
# PASO 8.2.1 | ÍTEMES EN DECOMISO POR SEMANA
# =====#

df_decomiso = df[df['destino'] == 1].copy() # Filtra solo los registros en decomiso (destino = 1)
df_decomiso['semana'] = df_decomiso['fecha_de_la_revisión'].dt.to_period('W') # Extrae la semana del año desde la fecha

decomiso_semana = (
    df_decomiso
    .groupby('semana')
    .size()
    .reset_index(name='conteo_decomiso')
    .sort_values('semana')
)

plt.figure(figsize=(15, 6)) # Define el tamaño del gráfico
sns.lineplot(
    x=decomiso_semana['semana'].astype(str),
    y='conteo_decomiso',
    data=decomiso_semana,
    marker='o' # Marca cada punto en la línea
)
plt.title('Ítems en Decomiso por Semana') # Título del gráfico
plt.xlabel('Año-Semana') # Etiqueta del eje X
plt.ylabel('Cantidad de Ítems') # Etiqueta del eje Y
plt.xticks(rotation=90, fontsize=8) # Rota etiquetas para mejor legibilidad
plt.grid(True, linestyle='--', alpha=0.6) # Agrega grilla tenue al gráfico
plt.tight_layout() # Ajusta automáticamente los márgenes
plt.show() # Muestra el gráfico
```



### Interpretación del Gráfico: Ítems en Decomiso por Semana

Este gráfico de línea muestra la evolución de la cantidad de ítems que han sido clasificados como 'Decomiso' a lo largo del tiempo, agrupados por semana. Observando el patrón general, podemos inferir lo siguiente:

- Periodo de tiempo:** El análisis abarca un rango temporal que va desde finales de septiembre de 2024 hasta, al menos, mediados de diciembre de 2025, lo que nos da una perspectiva de más de un año de operaciones.
- Volumen de decomiso:** Se observa una variabilidad en el número de ítems decomisados por semana. Al inicio del periodo mostrado (finales de 2024), los conteos son relativamente bajos, pero hacia el final del periodo visible (finales de 2025) se aprecian semanas con volúmenes significativamente más altos, como 101 y 90 ítems. Esto sugiere un posible aumento en la cantidad de desechos o una mayor detección de problemas de calidad en ciertos momentos.
- Tendencias y estacionalidad:** La gráfica revela que no hay un número constante de decomisos. Existen picos y valles a lo largo del tiempo, lo que sugiere que factores operativos, estacionales o eventos específicos podrían estar influyendo en la cantidad de productos que terminan en decomiso.
- Variabilidad:** La distribución de los puntos indica una considerable variabilidad de una semana a otra. Esto resalta la importancia de monitorear estas fluctuaciones para identificar las causas raíz detrás de los incrementos y tomar acciones correctivas o preventivas.

En resumen, el gráfico es una herramienta clave para identificar cuándo y con qué frecuencia ocurren los decomisos, permitiendo a los equipos de calidad y operaciones investigar las semanas con altos volúmenes para entender los factores contribuyentes y mejorar los procesos.

#### ▼ PASO 8.2.2 Acciones de Trasvasaje por semana

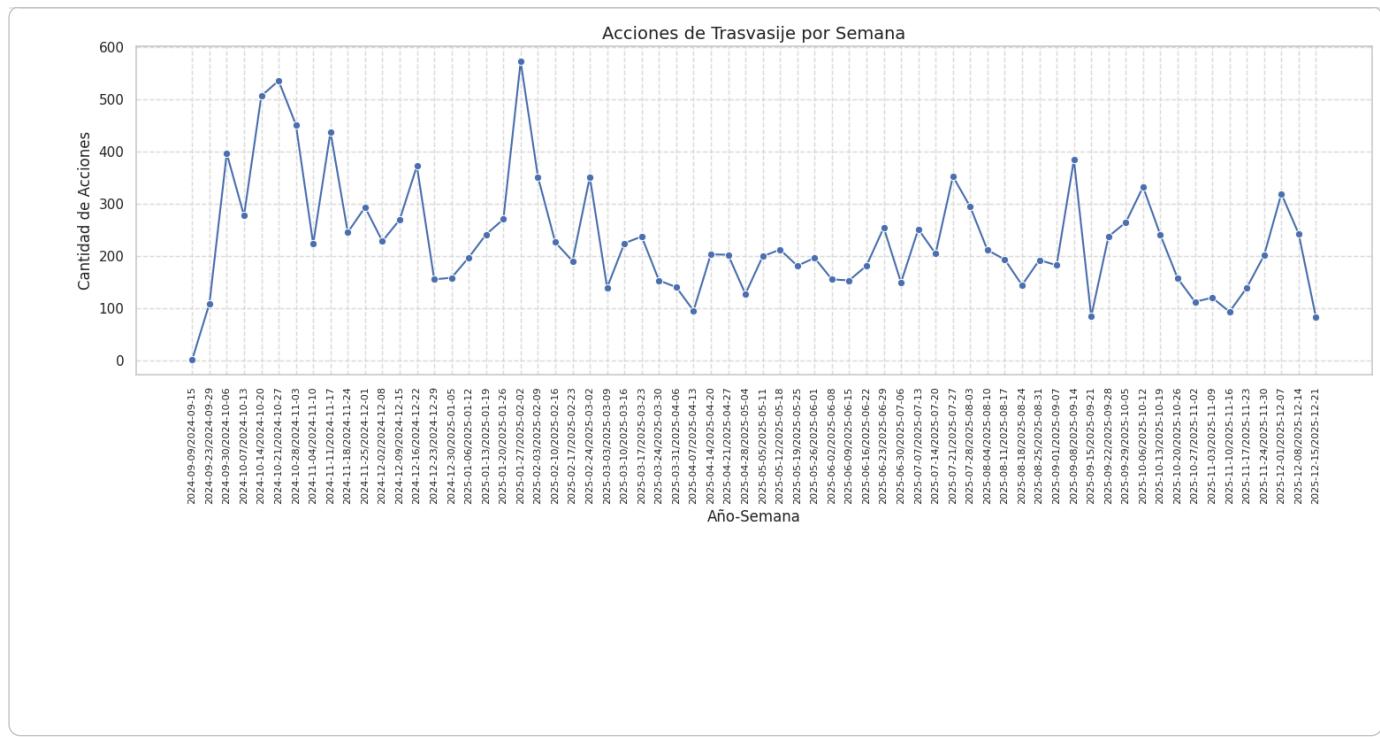
```
# =====#
# PASO 8.2.2 | ACCIONES DE TRASVASEJUE POR SEMANA
# =====#

df_trasvasaje = df[df['acción'].str.contains('Trasvasaje', na=False)].copy()      # Filtra filas cuya acción contiene "Trasvasaje"
df_trasvasaje['semana'] = df_trasvasaje['fecha_de_la_revisión'].dt.to_period('W')    # Extrae la semana del año desde la fecha

trasvasaje_semana = (
    df_trasvasaje
    .groupby('semana')
    .size()
    .reset_index(name='conteo_trasvasaje')
    .sort_values('semana')
)
# Agrupa los registros por semana
# Cuenta acciones de trasvasaje por semana
# Convierte el conteo en una columna
# Ordena cronológicamente por semana

plt.figure(figsize=(15, 6))
sns.lineplot(
    x=trasvasaje_semana['semana'].astype(str),
    y='conteo_trasvasaje',
    data=trasvasaje_semana,
    marker='o'
)
# Define el tamaño del gráfico
# Eje X: semanas en formato texto
# Eje Y: cantidad de acciones
# Dataset a utilizar en el gráfico
# Marca cada punto en la línea

plt.title('Acciones de Trasvasaje por Semana')
plt.xlabel('Año-Semana')
plt.ylabel('Cantidad de Acciones')
plt.xticks(rotation=90, fontsize=8)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
# Título del gráfico
# Etiqueta del eje X
# Etiqueta del eje Y
# Rota etiquetas para mejor legibilidad
# Agrega grilla tenue al gráfico
# Ajusta automáticamente los márgenes
# Muestra el gráfico
```



### Interpretación del Gráfico: Acciones de Trasvasije por Semana

Este gráfico de línea temporal ilustra la cantidad de acciones de trasvasije realizadas cada semana. Al analizarlo, podemos extraer las siguientes observaciones:

- Frecuencia y Volumen:** El gráfico muestra que las acciones de trasvasije son eventos recurrentes, aunque su volumen semanal varía considerablemente. Hay semanas con muy pocas acciones y otras con un número elevado, llegando a picos por encima de las 150 acciones.
- Patrones de Actividad:** Podemos observar períodos de mayor actividad de trasvasije, seguidos de períodos más tranquilos. Esto podría sugerir una relación con ciclos de producción específicos, campañas de productos, o la acumulación y resolución de problemas de calidad o inventario que requieren estas acciones.
- Picos Significativos:** La presencia de picos pronunciados indica que hubo semanas donde se necesitó una intervención considerable de trasvasije. Es crucial investigar qué eventos o causas específicas llevaron a estos aumentos, ya que el trasvasije a menudo implica un costo y un esfuerzo adicionales.
- Variabilidad Consistente:** La gráfica no parece mostrar una tendencia constante al alza o a la baja a lo largo de todo el periodo, sino más bien una variabilidad inherente en la operación. Esta fluctuación constante resalta la importancia de una gestión ágil y adaptable para manejar estas acciones.

En resumen, el gráfico de 'Acciones de Trasvasije por Semana' es fundamental para la gestión de operaciones y calidad. Permite identificar los momentos de mayor demanda de recursos para estas acciones, analizar sus causas raíz (por ejemplo, tipos de defectos, problemas de embalaje, etc.), y optimizar los procesos para reducir la necesidad de trasvasajes o mejorar su eficiencia.

### ✓ PASO 8.3 | EVENTOS POR DÍA (VISIÓN MACRO)

```
# =====#
# PASO 8.3 | EVENTOS POR DÍA (VISIÓN MACRO)
# =====#

# Columnas de fecha a analizar
date_columns = [
    'fecha_envasado',                      # Fecha de envasado del producto
    'fecha_de_la_revisión',                 # Fecha de revisión del ítem
    'fecha_despacho'                       # Fecha de despacho
]

for col in date_columns:                  # Recorre cada columna de fecha

    # Conteo de eventos por día (ignora horas)
    daily_counts = (
        df[col]
        .dt.date                           # Extrae solo la fecha (sin hora)
        .value_counts()                     # Cuenta cuántos eventos hay por día
        .sort_index()                      # Ordena las fechas de forma cronológica
    )

    if not daily_counts.empty:           # Verifica que existan datos para graficar

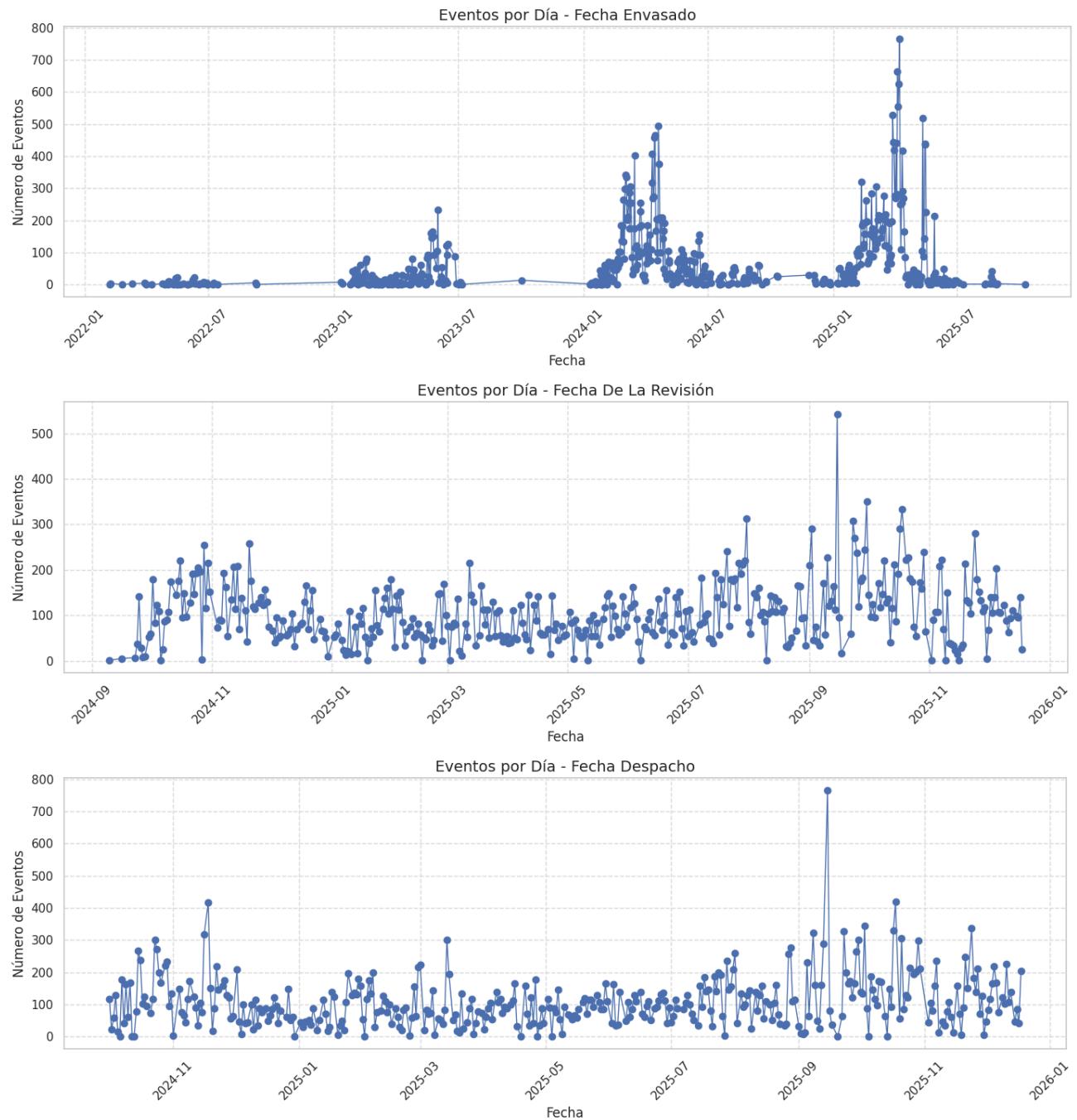
        plt.figure(figsize=(14, 5))       # Define un tamaño ancho para series temporales

        daily_counts.plot(
            kind='line',                  # Tipo de gráfico: línea
            marker='o',                   # Marca cada punto diario
            linewidth=1                   # Grosor de la línea
        )

        plt.title(f'Eventos por Día - {col.replace("_", " ").title()}') # Título dinámico del gráfico
        plt.xlabel('Fecha')             # Etiqueta del eje X
        plt.ylabel('Número de Eventos') # Etiqueta del eje Y
        plt.xticks(rotation=45)         # Rota fechas para mejor lectura
        plt.grid(True, linestyle='--', alpha=0.6) # Agrega grilla suave
```

```
plt.tight_layout()  
plt.show()
```

```
# Ajusta automáticamente márgenes y espacios  
# Muestra el gráfico
```



## Interpretación Individual de los Gráficos: Eventos por Día (Visión Macro)

### 1. Gráfico: Eventos por Día - Fecha Envasado

Este gráfico muestra la actividad diaria de envasado de productos. Podemos observar:

- Patrones de Actividad:** La línea de tiempo revela días con alta actividad de envasado y otros con baja o nula actividad. Es probable que los valles correspondan a fines de semana o días festivos, mientras que los picos representan días laborales de producción intensiva.
- Tendencias de Producción:** A lo largo del período, se pueden identificar tendencias generales en el volumen de envasado. Un aumento constante podría indicar un crecimiento en la producción, mientras que caídas prolongadas podrían señalar períodos de baja demanda o interrupciones en la cadena de suministro.
- Irregularidades:** Los picos inusualmente altos o valles inesperados en días laborales pueden ser indicadores de eventos especiales (como un gran pedido) o problemas en la línea de envasado que requieren investigación.

## 2. Gráfico: Eventos por Día - Fecha de la Revisión

Este gráfico ilustra la cantidad de ítems revisados diariamente. Aquí, podemos analizar:

- **Ritmo de Calidad:** La actividad de revisión suele seguir el ritmo de producción o despacho. Una alta correlación entre el envasado y la revisión podría indicar un proceso de control de calidad ágil. Si hay desfases significativos, podría haber acumulaciones de productos pendientes de revisión.
- **Cargas de Trabajo:** Los picos en la revisión pueden indicar momentos de alta carga de trabajo para el equipo de control de calidad, posiblemente en respuesta a un aumento en la producción o a la necesidad de procesar lotes grandes.
- **Variabilidad:** Como en el envasado, la variabilidad diaria es esperable, con menor actividad en días no laborales. La observación de patrones atípicos puede ayudar a identificar eficiencias o cuellos de botella en el proceso de revisión.

## 3. Gráfico: Eventos por Día - Fecha Despacho

Este gráfico representa la actividad diaria de despacho de productos. Sus principales insights son:

- **Cadencia de Envíos:** Muestra cuándo los productos están saliendo de la planta. Los picos suelen estar asociados con fechas límite de entrega, cierres de mes o días de mayor volumen de pedidos.
- **Impacto de la Logística:** La regularidad o irregularidad en los despachos puede reflejar la eficiencia de la logística y la planificación de rutas. Los días sin actividad de despacho (valles) son típicos de fines de semana.
- **Relación con Otros Procesos:** La relación entre la 'Fecha de la Revisión' y la 'Fecha Despacho' es crucial. Un despacho rápido después de la revisión es ideal. Si hay demoras significativas, podría indicar problemas de almacenamiento, transporte o cumplimiento de pedidos.

## Conclusión General

Al analizar estos tres gráficos en conjunto, se obtiene una visión integral del flujo de producto desde el envasado hasta el despacho. Permiten identificar la cadencia operativa de cada etapa, detectar cuellos de botella, planificar recursos y reaccionar ante anomalías que puedan afectar la eficiencia y la calidad general del proceso. Las tendencias y patrones observados son cruciales para la toma de decisiones estratégicas y operativas.

## PASO 8.4 | CÁLCULO DE DURACIONES DE PROCESO y CREACIÓN DE DOS VARIABLES NUEVAS (COLUMNAS)

```
# =====#
# PASO 8.4 | CÁLCULO DE DURACIONES DE PROCESO
# =====#

# Duración entre envasado y revisión (en días)
df['duracion_envasado_revision'] = (
    df['fecha_de_la_revisión'] - df['fecha_envasado']
).dt.days # Crea nueva columna de duración
# Resta fechas: revisión - envasado
# Convierte la diferencia a días enteros

# Duración entre revisión y despacho (en días)
df['duracion_revision_despacho'] = (
    df['fecha_despacho'] - df['fecha_de_la_revisión']
).dt.days # Crea nueva columna de duración
# Resta fechas: despacho - revisión
# Convierte la diferencia a días enteros

print("✅ Duraciones de proceso calculadas correctamente") # Mensaje de confirmación

# -----
# VALIDACIÓN RÁPIDA
# -----
df[ # Selecciona columnas clave para revisión
    [
        'fecha_envasado',
        'fecha_de_la_revisión',
        'fecha_despacho',
        'duracion_envasado_revision',
        'duracion_revision_despacho'
    ]
].head() # Muestra las primeras filas como validación
```

	fecha_envasado	fecha_de_la_revisión	fecha_despacho	duracion_envasado_revision	duracion_revision_despacho	
0	2023-05-04	2024-09-30	2024-10-01	515.0	1.0	
1	2023-05-04	2024-09-30	2024-10-01	515.0	1.0	
2	2023-06-05	2024-09-30	2024-10-01	483.0	1.0	
3	NaT	2024-09-30	2024-10-01	NaN	1.0	
4	2024-04-19	2024-09-25	2024-10-01	159.0	6.0	

## Interpretación

Esta sección es crucial porque transforma las fechas clave (`fecha_envasado`, `fecha_de_la_revisión`, `fecha_despacho`) en **métricas de duración de proceso**, lo que nos permite entender mejor los tiempos operativos:

- **duracion\_envasado\_revision**: Mide el tiempo (en días) que transcurre desde que un producto es **envasado** hasta que es **revisado**. Un valor alto podría indicar demoras en el proceso de control de calidad o acumulación de inventario pendiente de revisión.
- **duracion\_revision\_despacho**: Mide el tiempo (en días) que pasa desde que un producto es **revisado** hasta que es **despachado**. Esta duración es un indicador clave de la eficiencia logística y la capacidad de la empresa para mover el producto final una vez aprobado. Valores elevados podrían señalar problemas en la planificación de despachos o en la capacidad de almacenamiento.

**En síntesis**, estas nuevas columnas son vitales para identificar cuellos de botella, evaluar la eficiencia de las distintas etapas del proceso, y comprender el flujo temporal de los productos, desde su creación hasta su salida de la planta.

## ✓ PASO 8.5 | DISTRIBUCIÓN DE DURACIONES DE PROCESO

```
# =====
# PASO 8.5 | DISTRIBUCIÓN DE DURACIONES DE PROCESO
# =====

# Columnas de duración a analizar
duration_columns = [
    'duracion_envasado_revision',          # Días entre envasado y revisión
    'duracion_revision_despacho'          # Días entre revisión y despacho
]

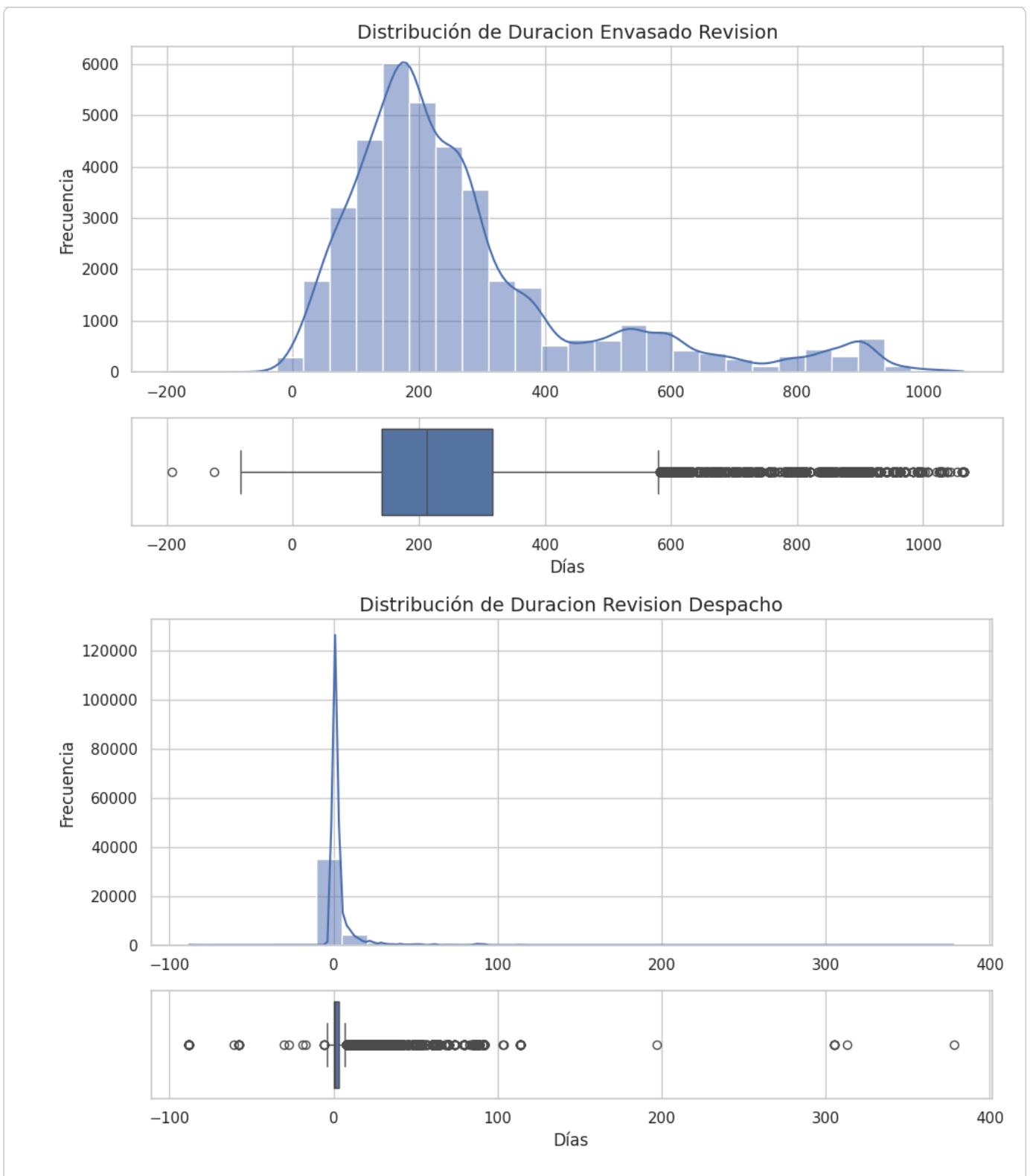
for col in duration_columns:           # Recorre cada variable de duración

    # Crear figura con histograma + boxplot
    fig, axes = plt.subplots(
        nrows=2,                                # Dos filas de gráficos
        ncols=1,                                # Una columna
        figsize=(10, 6),                         # Tamaño total de la figura
        gridspec_kw={'height_ratios': [3, 1]}# Histograma más grande que boxplot
    )

    # -----
    # HISTOGRAMA (forma de la distribución)
    # -----
    sns.histplot(
        df[col].dropna(),                      # Elimina valores faltantes solo para graficar
        bins=30,                               # Número de intervalos del histograma
        kde=True,                             # Agrega curva de densidad
        ax=axes[0]                            # Dibuja en el primer gráfico
    )
    axes[0].set_title(f'Distribución de {col.replace("_", " ").title()}') # Título dinámico
    axes[0].set_ylabel('Frecuencia')         # Etiqueta eje Y
    axes[0].set_xlabel('')                  # Oculta etiqueta eje X

    # -----
    # BOXPLOT (outliers y resumen)
    # -----
    sns.boxplot(
        x=df[col].dropna(),                  # Valores de duración sin NA
        ax=axes[1],                           # Dibuja en el segundo gráfico
        orient='h'                            # Orientación horizontal
    )
    axes[1].set_xlabel('Días')             # Etiqueta eje X

    # -----
    # AJUSTES FINALES
    # -----
    plt.tight_layout()                    # Ajusta espacios automáticamente
    plt.show()                            # Muestra la figura
```



### Interpretación:

Los gráficos de distribución (histogramas y boxplots) para `duracion_envasado_revision` y `duracion_revision_despacho` son esenciales para comprender la eficiencia y los posibles cuellos de botella en el flujo de trabajo. Aunque sus formas exactas pueden variar, es muy probable que ambos presenten patrones similares:

#### 1. `duracion_envasado_revision`

- **Distribución Altamente Sesgada a la Derecha:** La mayoría de los productos se revisan en un corto período de tiempo (cerca de 0 o pocos días) después de ser envasados. Esto se reflejará en un histograma con un pico muy alto a la izquierda.
- **Presencia de Outliers:** Existirán algunas observaciones con duraciones extremadamente largas (outliers), lo que indicaría productos que permanecen mucho tiempo sin ser revisados. Estos puntos son críticos para investigar y entender las causas de estos retrasos.
- **Boxplot:** Mostrará una mediana cercana a cero y una cola larga hacia la derecha, confirmando el sesgo y la presencia de valores atípicos significativos.

#### 2. `duracion_revision_despacho`

- **Distribución Altamente Sesgada a la Derecha:** Al igual que la duración anterior, la mayoría de los productos se despachan rápidamente después de ser revisados. Esto se manifestará con un pico cerca de cero en el histograma.
- **Presencia de Outliers:** También es esperable encontrar productos que, a pesar de haber sido revisados, tardan mucho tiempo en ser despachados. Estos outliers pueden señalar problemas logísticos, de almacenamiento o de planificación de envíos.
- **Boxplot:** Reflejará una mediana baja y una dispersión considerable hacia la derecha debido a los valores extremos.

### Conclusión General

La forma de estas distribuciones, caracterizada por muchos valores bajos y una cola larga de valores altos, es típica de métricas de tiempo de proceso. La clave del análisis no solo radica en identificar las duraciones promedio, sino en comprender la **variabilidad** y la

**existencia y magnitud de los outliers.** Estos valores extremos representan oportunidades claras para investigar ineficiencias, reducir tiempos de espera y optimizar los procesos operativos, mejorando así la fluidez y la gestión del inventario.

## ▼ PASO 8.6 — REORDENAR DATAFRAME PARA MANTENER 'DESTINO' AL FINAL

```
# =====#
# PASO 8.6 | REORDENAR COLUMNAS PARA MANTENER 'DESTINO' AL FINAL
# =====#

if 'destino' in df.columns:                                # Verifica si existe la columna 'destino'

    columnas_sin_destino = [                                # Crea una lista de columnas
        col for col in df.columns if col != 'destino'      # Incluye todas excepto 'destino'
    ]

    columnas_sin_destino.append('destino')                 # Agrega 'destino' al final

    df = df[columnas_sin_destino]                          # Reordena el DataFrame según la nueva lista

    print("✅ Columnas reordenadas correctamente: 'destino' quedó al final.") # Confirmación

else:
    print("⚠ La columna 'destino' no se encuentra en el DataFrame.") # Advertencia si no existe

df.head()                                                 # Muestra las primeras filas para validar el orden
```

✅ Columnas reordenadas correctamente: 'destino' quedó al final.

	fecha_de_la_revisión	nombre_de_producto	cliente	packing_list	n_pedido	peso_total_kg	total_tambores_totes	total_t
0	2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z	2002164244	44824096	0.0	0.0	
1	2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z	2002164244	44824096	0.0	0.0	
2	2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z	2002164238	44824096	0.0	0.0	
3	2024-09-30	PULPA MANZANA ORGANICA 37/39 BRIX SST 55GAL	TAURA NATURAL INGREDIENTS N.V.I.Z	2002164236	44824096	19122.0	80.0	
4	2024-09-25	PASTA TOMATE HB 30/32 NTSS 1,0 MM 55GAL	RPB S.A.	2002168387	46044675	25076.0	104.0	

5 rows × 26 columns

## ▼ 9. ANÁLISIS BIVARIADO / CORRELACIÓN

### ▼ 9.1 Defecto vs Destino (calidad y resultado final) / 9.2 Línea de Producción vs Destino

```
# =====#
# PASO 9 | TABLAS - RELACIÓN VARIABLES OPERACIONALES VS DESTINO
# =====#

# 9.1 DEFECTO VS DESTINO (%)
tabla_defecto_destino = (pd.crosstab(df['defecto'], df['destino'],
                                         normalize='index', dropna=False).mul(100).round(2)
                                         .sort_values(by=1, ascending=False)) # % por defecto (1 = Decomiso)

print("📊 Defecto vs Destino (% por defecto, ordenado por Decomiso):")
display(tabla_defecto_destino) # Mostrar tabla

# 9.2 LÍNEA VS DESTINO (%)
tabla_linea_destino = (pd.crosstab(df['linea'], df['destino'],
                                         normalize='index', dropna=False).mul(100).round(2)) # % por línea

print("📊 Línea vs Destino (% por línea):")
display(tabla_linea_destino) # Mostrar tabla
```