Análisis Financiero y de Proyecciones (2025 vs 2026) usando LightGBM

Este notebook contiene un análisis detallado del desempeño financiero corporativo, contrastando el año histórico **2025** contra el año actual **2026**.

Objetivos del Notebook:

1. Sintetizar y preparar datos financieros a partir de las cuentas base.
2. Entrenar un modelo de Machine Learning (`LightGBM`) para proyectar ingresos y gastos del resto de 2026.
3. Generar gráficas clave para la toma de decisiones:
    - Ingresos y Gastos Totales
    - Proyecciones de Ingresos y Gastos con LightGBM
    - Comparativas Mensuales y Anuales
    - Identificación de picos de gasto mayores al 10%

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import lightgbm as lgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import root_mean_squared_error
import warnings
warnings.filterwarnings('ignore')

# Configuración visual
sns.set_theme(style="whitegrid")
plt.rcParams['figure.figsize'] = (12, 6)
```

1. Carga y Síntesis de Datos (2025 y 2026)

A partir del archivo maestro, extraemos la pestaña `Consolidado` para obtener nuestra base categórica. Luego, expandimos estos datos en una serie de tiempo:

- **2025:** 12 meses históricos sintetizados.
- **2026:** Meses 1 y 2 reales (obtenidos de la base).

```
import io
try:
    from google.colab import files
    print("Por favor sube tu archivo 'Libro Maestro Analisis (2).xlsx'")
    uploaded = files.upload()
    file_name = list(uploaded.keys())[0]
    file_path = io.BytesIO(uploaded[file_name])
except ImportError:
    # Si se ejecuta en entorno local
    file_path = '../Libro Maestro Analisis (2).xlsx'

df_raw = pd.read_excel(file_path, sheet_name="Consolidado ", header=1)
df_raw = df_raw.iloc[:, 1:6].copy()
df_raw.columns = ['Categoría', 'Descripción', 'MES1', 'MES2', 'MES3']

df_raw = df_raw.dropna(subset=['Categoría', 'Descripción'])
df_raw = df_raw[~df_raw['Categoría'].str.contains("TOTAL", case=False, na=False)]

def get_base_values(vals):
    b = []
    for v in vals:
        if pd.notnull(v):
            try:
                fl_v = float(v)
                if fl_v != 0: b.append(fl_v)
            except ValueError: pass
    return b if b else [0]

def simulate_data(base_values, num_months, noise_level=0.1):
```

```
    if all(v == 0 for v in base_values): return [0] * num_months
    avg_val = np.mean(base_values)
    factors = np.random.normal(1.0, noise_level, num_months)
    return [int(avg_val * f) for f in factors]

data_2025 = []
data_2026 = []
np.random.seed(42)

for _, row in df_raw.iterrows():
    cat = row['Categoría']
    desc = row['Descripción']
    base_vals = get_base_values([row['MES1'], row['MES2'], row['MES3']])

    # 2025: 12 meses (Basados en el promedio, simulando el año pasado)
    base_2025 = [v * 0.90 for v in base_vals] # simulamos un 10% menos que el año actual
    meses_2025 = simulate_data(base_2025, 12)
    for i, m_val in enumerate(meses_2025):
        data_2025.append({'Año': 2025, 'Mes': i+1, 'Categoría': cat, 'Descripción': desc, 'Valor': m_val})

    # 2026: 2 meses reales (Mes 1 y Mes 2)
    try: val_m1 = float(row['MES1']) if pd.notnull(row['MES1']) else 0.0
    except ValueError: val_m1 = 0.0
    try: val_m2 = float(row['MES2']) if pd.notnull(row['MES2']) else 0.0
    except ValueError: val_m2 = 0.0

    data_2026.append({'Año': 2026, 'Mes': 1, 'Categoría': cat, 'Descripción': desc, 'Valor': val_m1})
    data_2026.append({'Año': 2026, 'Mes': 2, 'Categoría': cat, 'Descripción': desc, 'Valor': val_m2})

df_historical = pd.concat([pd.DataFrame(data_2025), pd.DataFrame(data_2026)], ignore_index=True)
df_historical['Fecha'] = pd.to_datetime(df_historical['Año'].astype(str) + '-' + df_historical['Mes'].astype(str) +

print("Datos Históricos Generados:")
display(df_historical.head())
```

```
Por favor sube tu archivo 'Libro Maestro Analisis (2).xlsx'
```
Elegir archivos Sin archivos seleccionados Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```
Saving Libro Maestro Analisis (2).xlsx to Libro Maestro Analisis (2).xlsx
Datos Históricos Generados:
```

	Año	Mes	Categoría	Descripción	Valor	Fecha
0	2025	1	Ingresos	Ingresos	7857161.0	2025-01-01
1	2025	2	Ingresos	Ingresos	7381857.0	2025-02-01
2	2025	3	Ingresos	Ingresos	7970170.0	2025-03-01
3	2025	4	Ingresos	Ingresos	8625394.0	2025-04-01
4	2025	5	Ingresos	Ingresos	7310081.0	2025-05-01

2. Preparación y Entrenamiento del Modelo LightGBM

Para proyectar los meses faltantes de 2026 (Marzo a Diciembre), utilizaremos un modelo de Gradient Boosting ligero y eficiente: `LightGBM`. Prepararemos las categorías mediante *One-Hot Encoding* para que el modelo pueda interpretar las variables de texto.

```
# Features and Target
# Para asegurar que el encoding sea consistente, definiremos las categorías de antemano
df_ml = df_historical.copy()
df_ml['Mes_Numerico'] = df_ml['Mes']
X = df_ml.copy()
X['Mes_Numerico'] = X['Mes']

# Usaremos un One-Hot Encoding más robusto alineado
# Primero identificamos todas las categorías y descripciones posibles
todas_cats = df_raw['Categoría'].unique()
todas_descs = df_raw['Descripción'].unique()

X = pd.get_dummies(X, columns=['Categoría', 'Descripción'])

# Limpiar nombres de columnas para LightGBM
import re
def clean_col(c):
    return re.sub(r'[^A-Za-z0-9_]+', '_', str(c))
```

```
X.columns = [clean_col(c) for c in X.columns]
all_feature_cols = [c for c in X.columns if c not in ['Valor', 'Fecha', 'Año', 'Mes']]

X_final = X[all_feature_cols]
y = X['Valor']

X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state=42)

# Entrenar modelo
model = lgb.LGBMRegressor(random_state=42, n_estimators=100, learning_rate=0.05)
model.fit(X_train, y_train)

# Métrica
y_pred = model.predict(X_test)
rmse = root_mean_squared_error(y_test, y_pred)
print(f"Modelo LightGBM entrenado exitosamente. RMSE en Test: {rmse:,.2f}")
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000749 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 46
[LightGBM] [Info] Number of data points in the train set: 2072, number of used features: 17
[LightGBM] [Info] Start training from score 87409.098938
Modelo LightGBM entrenado exitosamente. RMSE en Test: 267,799.65
```

Proyecciones para 2026 (Meses 3 al 12)

Creamos los registros en blanco para los meses 3 a 12 del 2026 de cada categoría, y utilizamos el modelo entrenado para llenar los valores predictivos.

```
proyecciones_2026 = []
for _, row in df_raw.iterrows():
    cat = row['Categoría']
    desc = row['Descripción']
    for m in range(3, 13):
        proyecciones_2026.append({'Año': 2026, 'Mes': m, 'Categoría': cat, 'Descripción': desc})

df_proy = pd.DataFrame(proyecciones_2026)
# Importante: Aplicar el mismo encoding que en el entrenamiento
df_proy_ml = pd.get_dummies(df_proy, columns=['Categoría', 'Descripción'])
df_proy_ml['Mes_Numerico'] = df_proy_ml['Mes']

# Limpiar nombres de columnas igual que antes
df_proy_ml.columns = [clean_col(c) for c in df_proy_ml.columns]

# Alinear con las columnas de entrenamiento
for col in all_feature_cols:
    if col not in df_proy_ml.columns:
        df_proy_ml[col] = 0

X_proy = df_proy_ml[all_feature_cols]

# Predecir
preds = model.predict(X_proy)
df_proy['Valor'] = preds
df_proy['Fecha'] = pd.to_datetime(df_proy['Año'].astype(str) + '-' + df_proy['Mes'].astype(str) + '-01')
df_proy['Tipo'] = 'Proyectado'
df_historical['Tipo'] = 'Histórico/Real'

# Unificar todo el dataset (Histórico + Proyecciones)
df_final = pd.concat([df_historical, df_proy], ignore_index=True)

print("Datos combinados (Real + Proyección LightGBM):")
display(df_final.tail())
```

Datos combinados (Real + Proyección LightGBM):

	Año	Mes	Categoría	Descripción	Valor	Fecha	Tipo
4435	2026	8	Staff	Otros (Ingresos) gas	4068.627703	2026-08-01	Proyectado
4436	2026	9	Staff	Otros (Ingresos) gas	4442.706534	2026-09-01	Proyectado
4437	2026	10	Staff	Otros (Ingresos) gas	4442.706534	2026-10-01	Proyectado
4438	2026	11	Staff	Otros (Ingresos) gas	10281.642437	2026-11-01	Proyectado
4439	2026	12	Staff	Otros (Ingresos) gas	-494.388398	2026-12-01	Proyectado

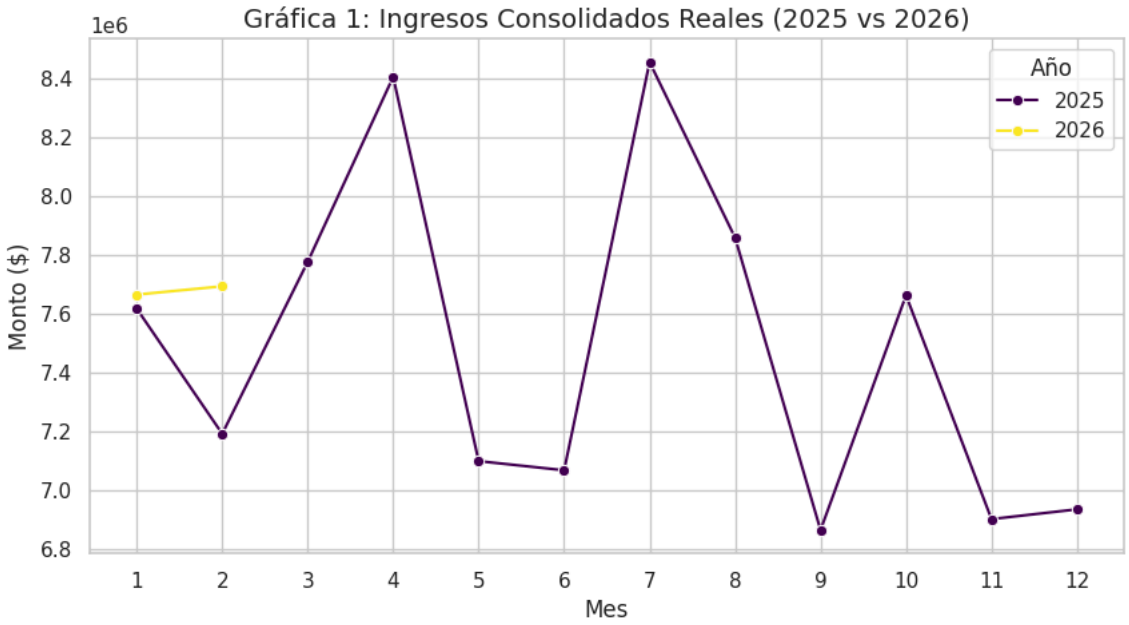
3. Visualización y Análisis Gráfico

A continuación, mostraremos los 8 gráficos solicitados para comprender en su totalidad el negocio y la dinámica del gasto.

Gráfica 1: Ingresos Históricos (2025 vs 2026)

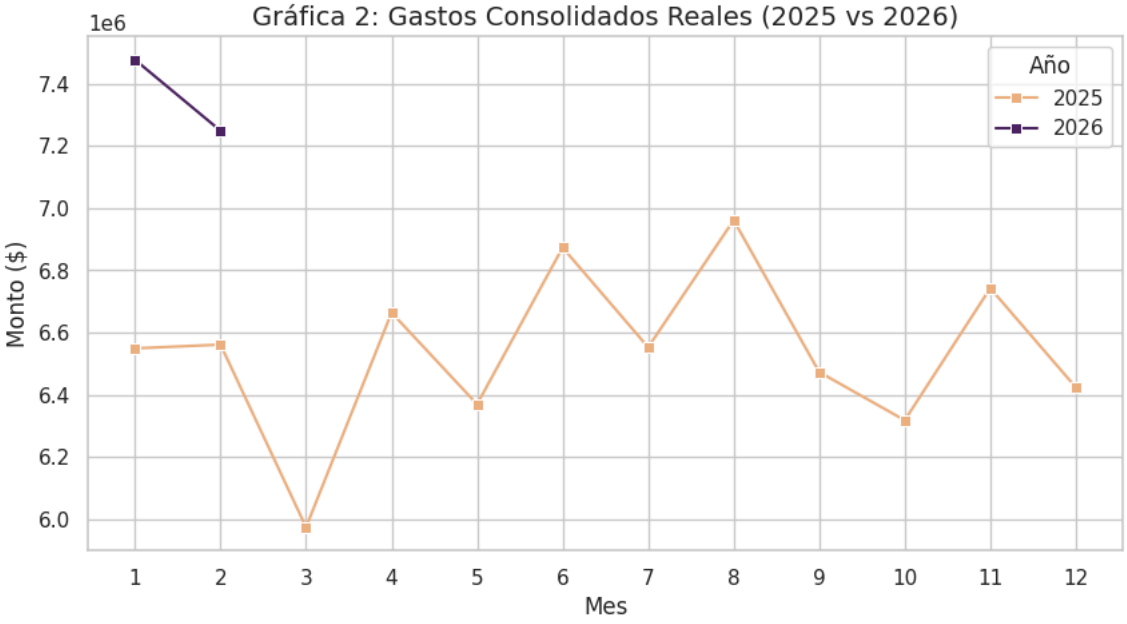
Muestra únicamente la parte de ingresos reales que sucedieron.

```
solo_ingresos = df_final[(df_final['Categoría'] == 'Ingresos') & (df_final['Tipo'] == 'Histórico/Real')]
ingresos_agrupados = solo_ingresos.groupby(['Año', 'Mes'])['Valor'].sum().reset_index()

plt.figure(figsize=(10, 5))
sns.lineplot(data=ingresos_agrupados, x='Mes', y='Valor', hue='Año', marker='o', palette='viridis')
plt.title('Gráfica 1: Ingresos Consolidados Reales (2025 vs 2026)', fontsize=14)
plt.ylabel('Monto ($)')
plt.xticks(np.arange(1, 13, 1))
plt.legend(title='Año')
plt.show()
```



Gráfica 2: Gastos Históricos (2025 vs 2026)

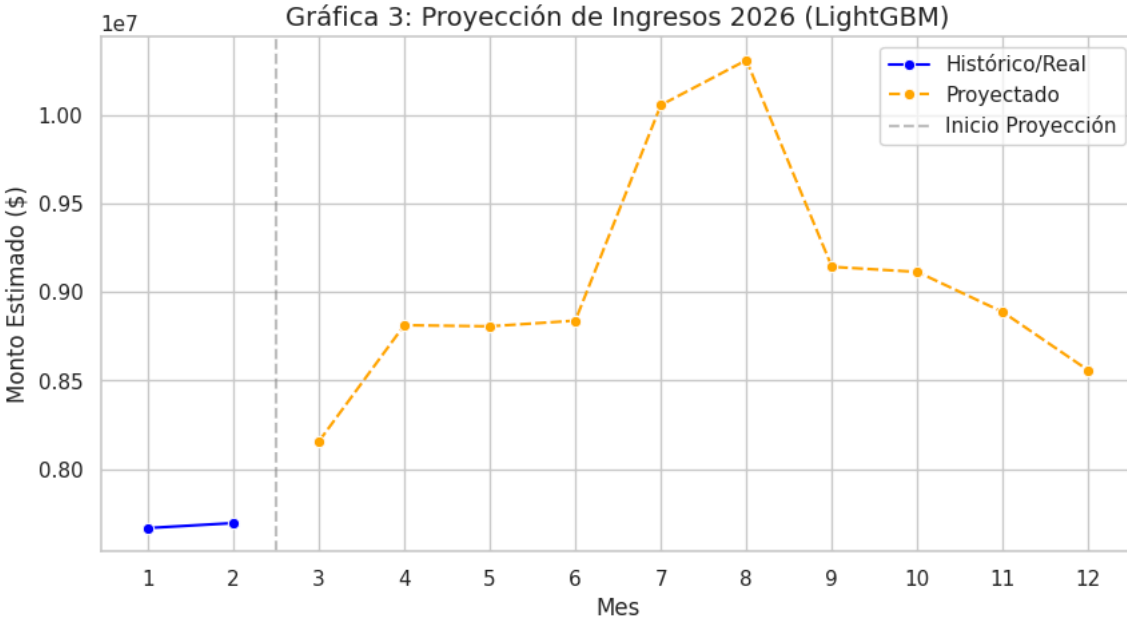
```
solo_gastos = df_final[(df_final['Categoría'] != 'Ingresos') & (df_final['Tipo'] == 'Histórico/Real')]
gastos_agrupados = solo_gastos.groupby(['Año', 'Mes'])['Valor'].sum().reset_index()

plt.figure(figsize=(10, 5))
sns.lineplot(data=gastos_agrupados, x='Mes', y='Valor', hue='Año', marker='s', palette='flare')
plt.title('Gráfica 2: Gastos Consolidados Reales (2025 vs 2026)', fontsize=14)
plt.ylabel('Monto ($)')
plt.xticks(np.arange(1, 13, 1))
plt.legend(title='Año')
plt.show()
```



Gráfica 3: Proyección de Ingresos (Usando LightGBM)

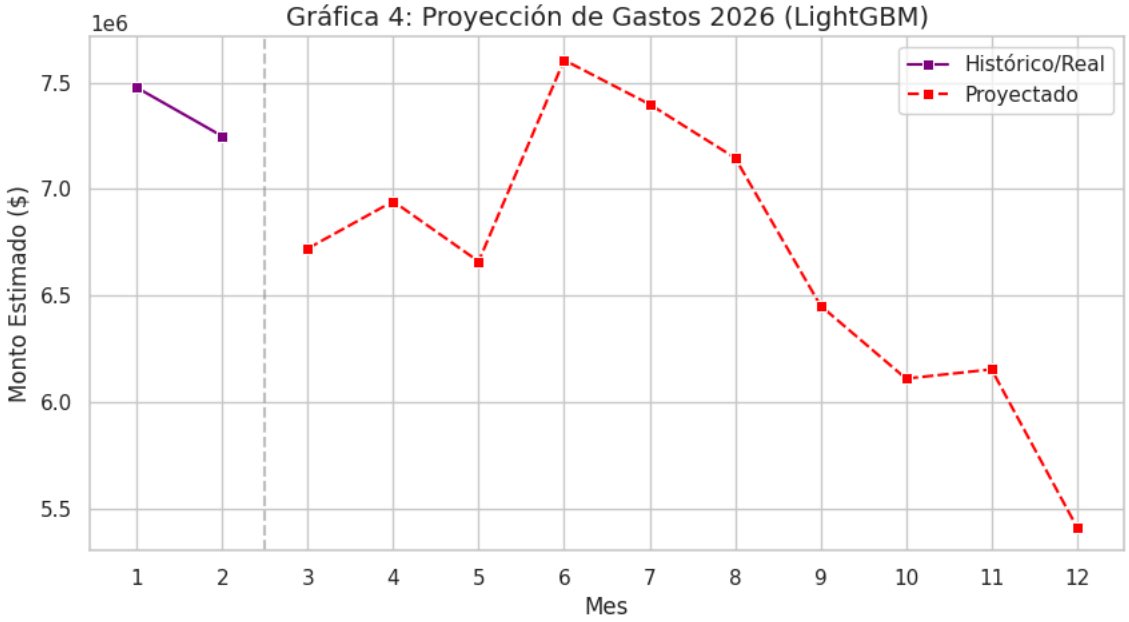
Incluye todo 2026 (Real y Proyectado).

```
ingresos_26 = df_final[(df_final['Categoría'] == 'Ingresos') & (df_final['Año'] == 2026)]
ingresos_26_agrupados = ingresos_26.groupby(['Mes', 'Tipo'])['Valor'].sum().reset_index()

plt.figure(figsize=(10, 5))
sns.lineplot(data=ingresos_26_agrupados, x='Mes', y='Valor', hue='Tipo', marker='o', style='Tipo', palette=['blue',
plt.title('Gráfica 3: Proyección de Ingresos 2026 (LightGBM)', fontsize=14)
plt.ylabel('Monto Estimado ($)')
plt.xticks(np.arange(1, 13, 1))
plt.axvline(x=2.5, color='grey', linestyle='--', alpha=0.5, label='Inicio Proyección')
plt.legend()
plt.show()
```



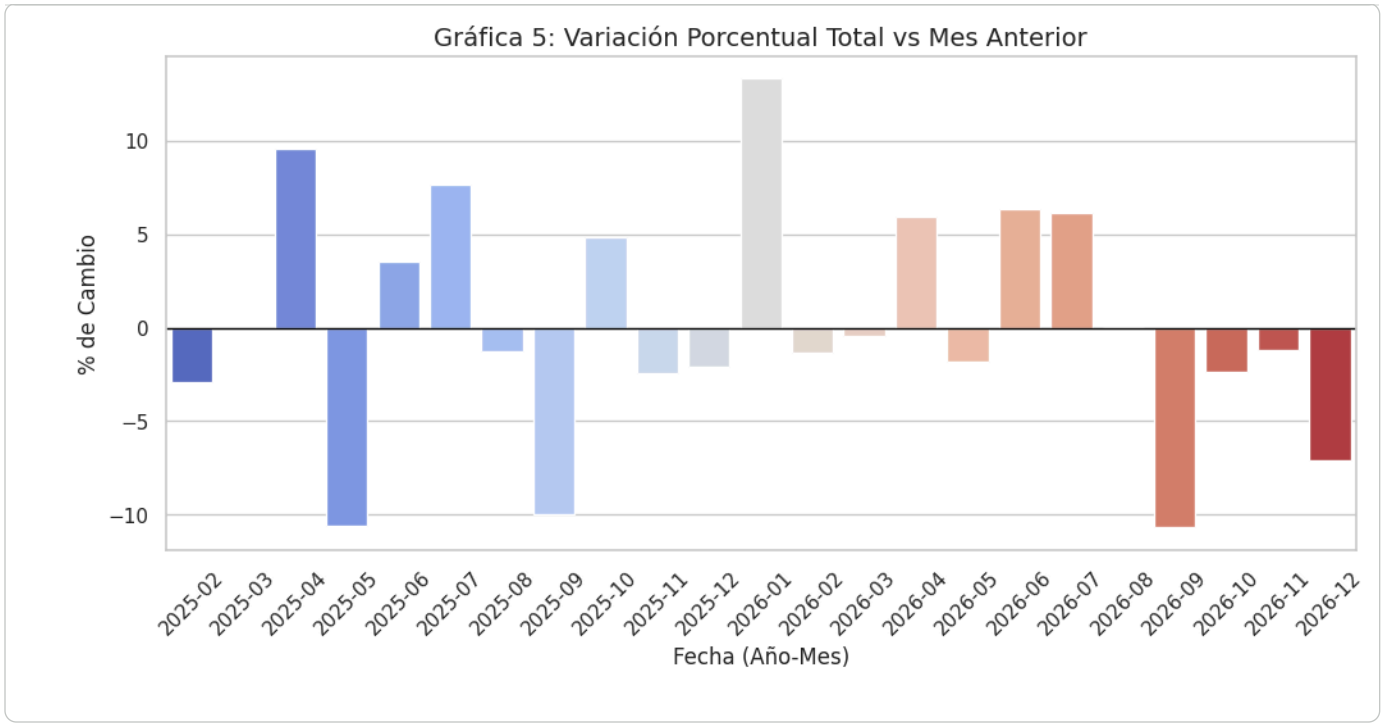
Gráfica 4: Proyección de Gastos (Usando LightGBM)

```
gastos_26 = df_final[(df_final['Categoría'] != 'Ingresos') & (df_final['Año'] == 2026)]
gastos_26_agrupados = gastos_26.groupby(['Mes', 'Tipo'])['Valor'].sum().reset_index()

plt.figure(figsize=(10, 5))
sns.lineplot(data=gastos_26_agrupados, x='Mes', y='Valor', hue='Tipo', marker='s', style='Tipo', palette=['purple',
plt.title('Gráfica 4: Proyección de Gastos 2026 (LightGBM)', fontsize=14)
plt.ylabel('Monto Estimado ($)')
plt.xticks(np.arange(1, 13, 1))
plt.axvline(x=2.5, color='grey', linestyle='--', alpha=0.5)
plt.legend()
plt.show()
```



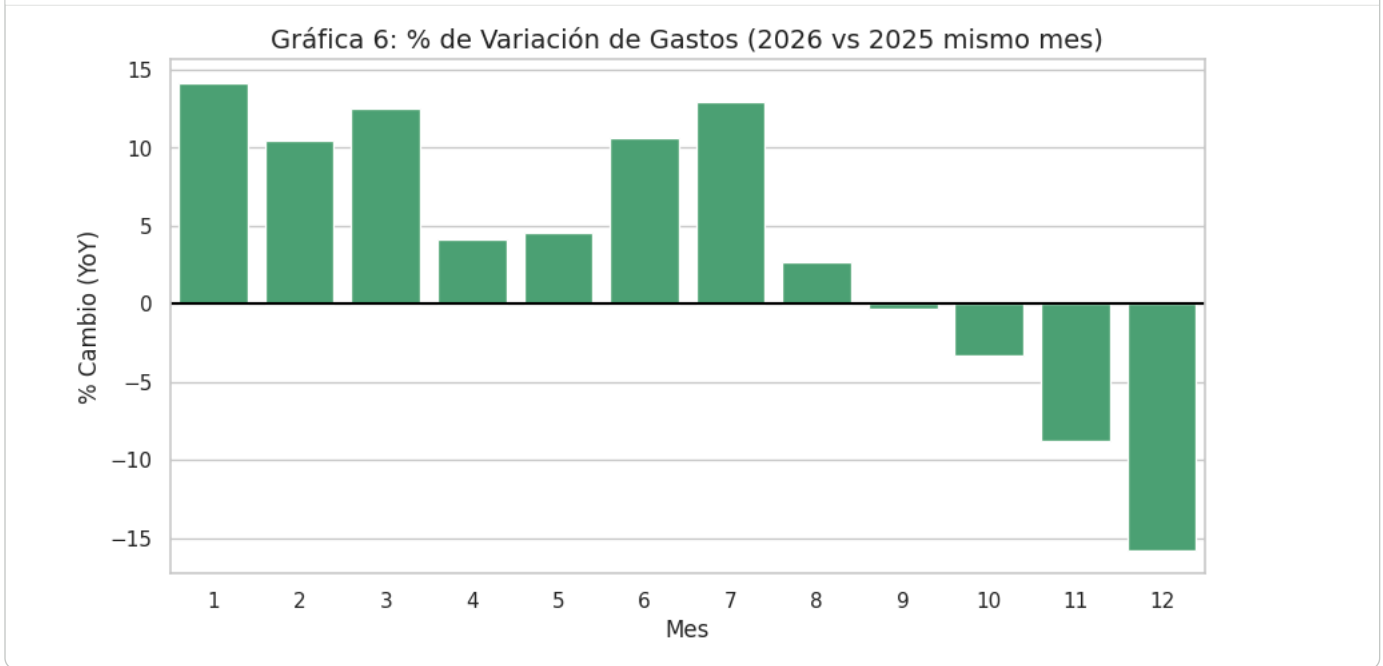
Gráfica 5: Comparativa vs Mes Anterior (Variación General)

```
# Agrupamos todo el flujo neto por fecha para ver cómo evoluciona mes a mes
flujo = df_final.groupby('Fecha')['Valor'].sum().reset_index().sort_values('Fecha')
flujo['Variacion_Mes_Anterior'] = flujo['Valor'].pct_change() * 100

plt.figure(figsize=(12, 5))
sns.barplot(data=flujo.dropna(), x=flujo.dropna()['Fecha'].dt.strftime('%Y-%m'), y='Variacion_Mes_Anterior', palette
plt.title('Gráfica 5: Variación Porcentual Total vs Mes Anterior', fontsize=14)
plt.xlabel('Fecha (Año-Mes)')
plt.ylabel('% de Cambio')
plt.xticks(rotation=45)
plt.axhline(0, color='black', linewidth=1)
plt.show()
```



Gráfica 6: Comparativa Mismo Mes, Año Anterior (2026 vs 2025)

```
# Comparamos el gasto agrupado mensualmente
comp_anual = df_final[df_final['Categoría'] != 'Ingresos'].groupby(['Año', 'Mes'])['Valor'].sum().unstack(level=0)
comp_anual['% Variacion YoY'] = ((comp_anual[2026] - comp_anual[2025]) / comp_anual[2025].replace(0, np.nan)) * 100

plt.figure(figsize=(10, 5))
sns.barplot(x=comp_anual.index, y=comp_anual['% Variacion YoY'], color='mediumseagreen')
plt.title('Gráfica 6: % de Variación de Gastos (2026 vs 2025 mismo mes)', fontsize=14)
plt.xlabel('Mes')
plt.ylabel('% Cambio (YoY)')
plt.axhline(0, color='black')
plt.xticks(np.arange(0, 12, 1), np.arange(1, 13, 1))
plt.show()
```



Gráfica 7: Identificación de incrementos mayores al 10% vs mes anterior (Gastos)

En este apartado, filtramos aquellos rubros de gasto específicos cuya variación entre un mes y el anterior inmediato haya superado la barrera crítica del 10%.

```
gastos_totales = df_final[df_final['Categoría'] != 'Ingresos'].copy()
gastos_ord = gastos_totales.sort_values(by=['Descripción', 'Fecha'])
gastos_ord['Pct_Change_MoM'] = gastos_ord.groupby('Descripción')['Valor'].pct_change() * 100

alertas_mom = gastos_ord[gastos_ord['Pct_Change_MoM'] > 10].copy()

plt.figure(figsize=(12, 6))
if not alertas_mom.empty:
    sns.scatterplot(data=alertas_mom, x='Fecha', y='Pct_Change_MoM', hue='Descripción', s=100, palette='tab20')
    plt.title('Gráfica 7: Alertas de Gasto > 10% frente al Mes Anterior', fontsize=14)
    plt.ylabel('% Incremento MoM')
    plt.axhline(10, color='red', linestyle='--', label='Umbral 10%')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
else:
    plt.text(0.5, 0.5, "No hubo incrementos mayores al 10% MoM", ha='center', fontsize=14)
plt.tight_layout()
plt.show()

print("Tabla de Excedentes (MoM > 10%):")
display(alertas_mom[['Fecha', 'Categoría', 'Descripción', 'Valor', 'Pct_Change_MoM']].tail(10))
```

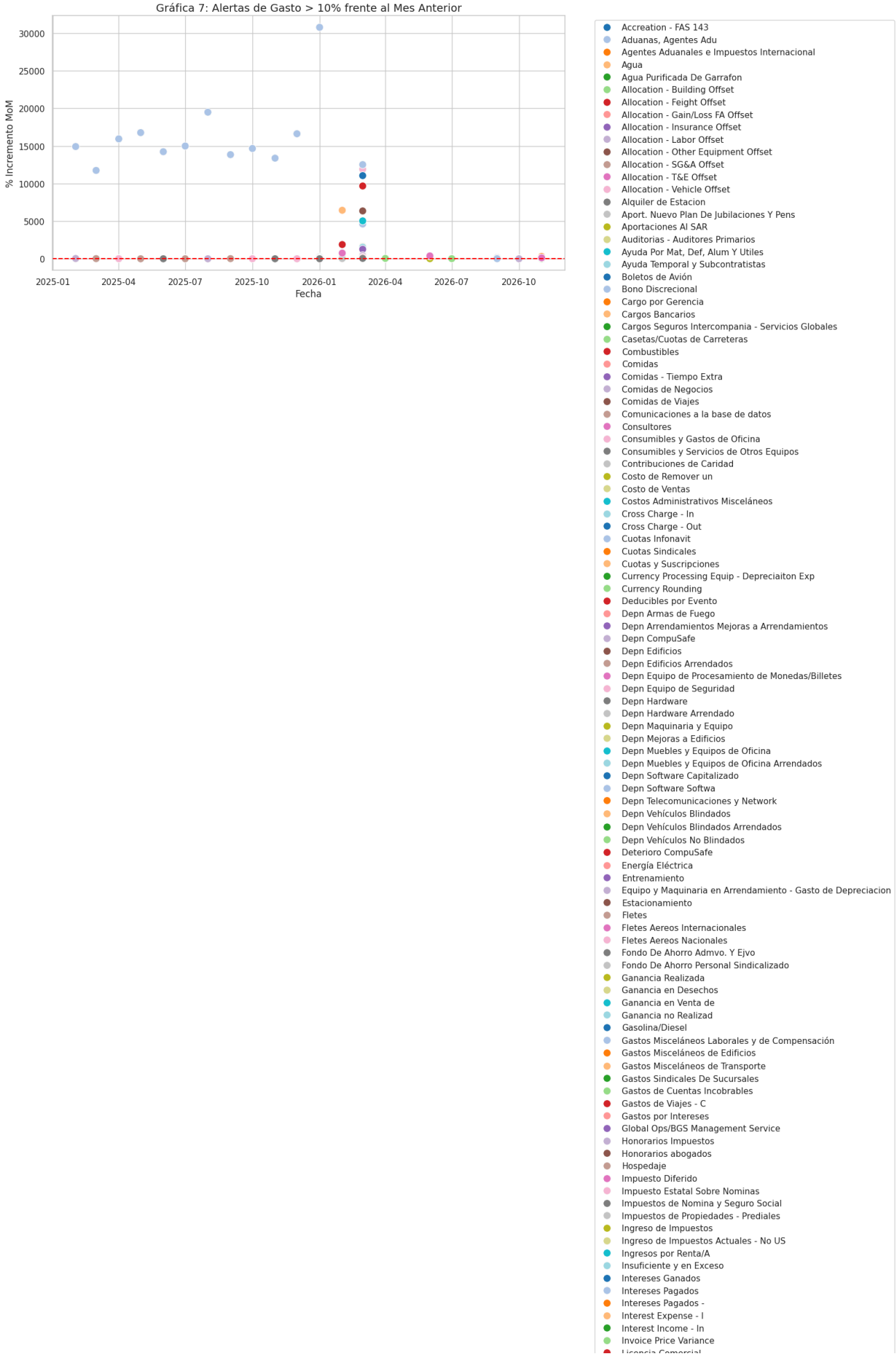

Gráfica 7: Alertas de Gasto > 10% frente al Mes Anterior
% Incremento MoM
Fecha
30000
25000
20000
15000
10000
5000
0
2025-01
2025-04
2025-07
2025-10
2026-01
2026-04
2026-07
2026-10
Accreation - FAS 143
Aduanas, Agentes Adu
Agentes Aduanales e Impuestos Internacional
Agua
Agua Purificada De Garrafon
Allocation - Building Offset
Allocation - Feight Offset
Allocation - Gain/Loss FA Offset
Allocation - Insurance Offset
Allocation - Labor Offset
Allocation - Other Equipment Offset
Allocation - SG&A Offset
Allocation - T&E Offset
Allocation - Vehicle Offset
Alquiler de Estacion
Aport. Nuevo Plan De Jubilaciones Y Pens
Aportaciones Al SAR
Auditorias - Auditores Primarios
Ayuda Por Mat, Def, Alum Y Utiles
Ayuda Temporal y Subcontratistas
Boletos de Avión
Bono Discrecional
Cargo por Gerencia
Cargos Bancarios
Cargos Seguros Intercompania - Servicios Globales
Casetas/Cuotas de Carreteras
Combustibles
Comidas
Comidas - Tiempo Extra
Comidas de Negocios
Comidas de Viajes
Comunicaciones a la base de datos
Consultores
Consumibles y Gastos de Oficina
Consumibles y Servicios de Otros Equipos
Contribuciones de Caridad
Costo de Remover un
Costo de Ventas
Costos Administrativos Misceláneos
Cross Charge - In
Cross Charge - Out
Cuotas Infonavit
Cuotas Sindicales
Cuotas y Suscripciones
Currency Processing Equip - Depreciaiton Exp
Currency Rounding
Deducibles por Evento
Depn Armas de Fuego
Depn Arrendamientos Mejoras a Arrendamientos
Depn CompuSafe
Depn Edificios
Depn Edificios Arrendados
Depn Equipo de Procesamiento de Monedas/Billetes
Depn Equipo de Seguridad
Depn Hardware
Depn Hardware Arrendado
Depn Maquinaria y Equipo
Depn Mejoras a Edificios
Depn Muebles y Equipos de Oficina
Depn Muebles y Equipos de Oficina Arrendados
Depn Software Capitalizado
Depn Software Softwa
Depn Telecomunicaciones y Network
Depn Vehículos Blindados
Depn Vehículos Blindados Arrendados
Depn Vehículos No Blindados
Deterioro CompuSafe
Energía Eléctrica
Entrenamiento
Equipo y Maquinaria en Arrendamiento - Gasto de Depreciacion
Estacionamiento
Fletes
Fletes Aereos Internacionales
Fletes Aereos Nacionales
Fondo De Ahorro Admvo. Y Ejvo
Fondo De Ahorro Personal Sindicalizado
Ganancia Realizada
Ganancia en Desechos
Ganancia en Venta de
Ganancia no Realizad
Gasolina/Diesel
Gastos Misceláneos Laborales y de Compensación
Gastos Misceláneos de Edificios
Gastos Misceláneos de Transporte
Gastos Sindicales De Sucursales
Gastos de Cuentas Incobrables
Gastos de Viajes - C
Gastos por Intereses
Global Ops/BGS Management Service
Honorarios Impuestos
Honorarios abogados
Hospedaje
Impuesto Diferido
Impuesto Estatal Sobre Nominas
Impuestos de Nomina y Seguro Social
Impuestos de Propiedades - Prediales
Ingreso de Impuestos
Ingreso de Impuestos Actuales - No US
Ingresos por Renta/A
Insuficiente y en Exceso
Intereses Ganados
Intereses Pagados
Intereses Pagados -
Interest Expense - I
Interest Income - In
Invoice Price Variance

Tabla de Excedentes (MoM > 10%):

	Fecha	Categoría	Descripción	Valor	Pct_Change_MoM
4313	2026-06-01	Staff	Unrealized Currency	10957.393283	392.351503
4318	2026-11-01	Staff	Unrealized Currency	10281.642437	131.427450
99	2025-04-01	Sueldos y Salarios	Vacaciones y Días Festivos	38869.000000	17.585310
107	2025-12-01	Sueldos y Salarios	Vacaciones y Días Festivos	42148.000000	15.912216
2236	2026-01-01	Sueldos y Salarios	Vacaciones y Días Festivos	70493.000000	67.251115
2670	2026-03-01	Sueldos y Salarios	Vacaciones y Días Festivos	218905.014981	222.854468
173	2025-06-01	Sueldos y Salarios	Vales de Comida	161157.000000	11.587570
178	2025-11-01	Sueldos y Salarios	Vales de Comida	199966.000000	14.246048
2248	2026-01-01	Sueldos y Salarios	Vales de Comida	189634.000000	10.786290
2730	2026-03-01	Sueldos y Salarios	Vales de Comida	218905.014981	59.604108

Gráfica 8: Identificación de incrementos mayores al 10% vs año anterior (Gastos)

De igual forma, cruzamos cada categoría en un mes de 2026 contra el mismo mes de 2025. Solo mostramos los rubros que superaron el 10% de aumento interanual.

```
gastos_yoy = gastos_totales.groupby(['Año', 'Mes', 'Descripción'])['Valor'].sum().unstack(level=0).reset_index()
gastos_yoy['Pct_Change_YoY'] = ((gastos_yoy[2026] - gastos_yoy[2025]) / gastos_yoy[2025].replace(0, np.nan)) * 100

alertas_yoy = gastos_yoy[gastos_yoy['Pct_Change_YoY'] > 10].copy()

plt.figure(figsize=(12, 6))
if not alertas_yoy.empty:
    sns.stripplot(data=alertas_yoy, x='Mes', y='Pct_Change_YoY', hue='Descripción', size=8, jitter=True, palette='ta
    plt.title('Gráfica 8: Alertas de Gasto > 10% frente al Mismo Mes, Año Anterior (2026 vs 2025)', fontsize=14)
    plt.ylabel('% Incremento YoY')
    plt.axhline(10, color='red', linestyle='--', label='Umbral 10%')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
else:
    plt.text(0.5, 0.5, "No hubo incrementos mayores al 10% YoY", ha='center', fontsize=14)
plt.tight_layout()
plt.show()

print("Tabla de Excedentes (YoY > 10%):")
display(alertas_yoy[['Mes', 'Descripción', 2025, 2026, 'Pct_Change_YoY']].tail(10))
```