Análisis de Datos:

Ventas de Tienda E-commerce

Autor: **Licdo. Rafael Jiménez**

Fecha: **01 de Octubre de 2025**

```
# Celda 1 Instalar numpy, pandas, matplotlib, seaborn
!pip install numpy

# Celda 2: Importar librerías
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Celda 3: Subir el archivo CSV
from google.colab import files
uploaded = files.upload()
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.
Elegir archivos Tienda Online.csv
• Tienda Online.csv(text/csv) - 3441207 bytes, last modified: 30/9/2025 - 100% done
Saving Tienda Online.csv to Tienda Online.csv
```

Introducción

Objetivos del Estudio

Este informe tiene como objetivo analizar los patrones de ventas de la empresa "XYZ Corp" durante el año 2023, identificando tendencias, estacionalidad y factores que influyen en el desempeño comercial.

Contexto del Problema

La empresa ha experimentado fluctuaciones en sus ventas y requiere un análisis detallado para:

- Identificar meses de alto y bajo desempeño
- Analizar el comportamiento por categorías de productos
- Proporcionar recomendaciones para optimizar ventas

```
# Lee el archivo subido (solo una de estas líneas)
import io
df = pd.read_csv(io.BytesIO(uploaded['Tienda Online.csv']), sep=",", encoding='iso-8859-1')

print("  Archivo cargado exitosamente!")

# Celda 5: Análisis del archivo, cuantas filas y columnas
print(f"El archivo tiene {df.shape[0]} filas y {df.shape[1]} columnas")
```

```
  Archivo cargado exitosamente!
El archivo tiene 34500 filas y 17 columnas
```

Haz doble clic (o pulsa Intro) para editar

```
# Después de cargar el archivo, siempre haz esto:
print("?? INSPECCIÓN INICIAL:")
print(f"?? Dimensiones: {df.shape[0]} filas, {df.shape[1]} columnas")
print("\n?? Primeras 7 filas:")
print(df.head(7))
print("\n?? Información general:")
print(df.info(7))
```

```
print("\n? Valores faltantes:")
print(df.isnull().sum())
```

```
5    Credit Card  2024-04-14                  5  South       No
6         PayPal  2025-05-20                  5   East       No

   total_amount  shipping_cost  profit_margin  customer_age customer_gender
0        139.47           7.88          31.17            60          Female
1         24.73           4.60          -2.62            37            Male
2        166.80           6.58          13.44            34            Male
3         63.67           5.50           2.14            21          Female
4         13.88           2.74           1.15            39            Male
5         97.04           6.32          37.35            35          Female
6        266.50           9.10          22.88            49            Male

?? Información general:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34500 entries, 0 to 34499
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   order_id            34500 non-null  object
 1   customer_id         34500 non-null  object
 2   product_id          34500 non-null  object
 3   category            34500 non-null  object
 4   price               34500 non-null  float64
 5   discount            34500 non-null  float64
 6   quantity            34500 non-null  int64
 7   payment_method      34500 non-null  object
 8   order_date          34500 non-null  object
 9   delivery_time_days  34500 non-null  int64
 10  region              34500 non-null  object
 11  returned            34500 non-null  object
 12  total_amount        34500 non-null  float64
 13  shipping_cost       34500 non-null  float64
 14  profit_margin       34500 non-null  float64
 15  customer_age        34500 non-null  int64
 16  customer_gender     34500 non-null  object
dtypes: float64(5), int64(3), object(9)
memory usage: 4.5+ MB
None

? Valores faltantes:
order_id              0
customer_id           0
product_id            0
category              0
price                 0
discount              0
quantity              0
payment_method        0
order_date            0
delivery_time_days    0
region                0
returned              0
total_amount          0
shipping_cost         0
profit_margin         0
customer_age          0
customer_gender       0
dtype: int64
```

```
# Convertir order_date a datetime
df['order_date'] = pd.to_datetime(df['order_date'])
```

```
# Después de cargar el archivo, siempre haz esto:
print("?? INSPECCIÓN INICIAL:")
print(f"?? Dimensiones: {df.shape[0]} filas, {df.shape[1]} columnas")
print("\n?? Primeras 7 filas:")
print(df.head(7))
print("\n?? Información general:")
print(df.info(7))
print("\n? Valores faltantes:")
print(df.isnull().sum())
```

```
3           5.50           2.14            21         Female
4           2.74           1.15            39           Male
5           6.32          37.35            35         Female
6           9.10          22.88            49           Male

?? Información general:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34500 entries, 0 to 34499
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   order_id            34500 non-null  object
 1   customer_id         34500 non-null  object
 2   product_id          34500 non-null  object
 3   category            34500 non-null  object
 4   price               34500 non-null  float64
 5   discount            34500 non-null  float64
 6   quantity            34500 non-null  int64
 7   payment_method      34500 non-null  object
 8   order_date          34500 non-null  datetime64[ns]
 9   delivery_time_days  34500 non-null  int64
 10  region              34500 non-null  object
 11  returned            34500 non-null  object
 12  total_amount        34500 non-null  float64
 13  shipping_cost       34500 non-null  float64
 14  profit_margin       34500 non-null  float64
 15  customer_age        34500 non-null  int64
 16  customer_gender     34500 non-null  object
dtypes: datetime64[ns](1), float64(5), int64(3), object(8)
memory usage: 4.5+ MB
None

? Valores faltantes:
order_id              0
customer_id           0
product_id            0
category              0
price                 0
discount              0
quantity              0
payment_method        0
order_date            0
delivery_time_days    0
region                0
returned              0
total_amount          0
shipping_cost         0
profit_margin         0
customer_age          0
customer_gender       0
dtype: int64
```

```
# Análisis Exploratorio
# Estadísticas descriptivas
print("?? Estadísticas descriptivas:")
print(df[['price', 'discount', 'quantity', 'total_amount', 'profit_margin', 'shipping_cost']].describe())

# Análisis por categorías
#print("\n?? Ventas por tipo de orden:")
#print(df['OrderType'].value_counts())

#print("\n?? Ventas por tipo de cliente:")
#print(df['CustomerType'].value_counts())

#print("\n?? Ventas por categoría de producto:")
#print(df['ProductCategory'].value_counts())

#print("\n?? Top 5 estados por ventas:")
#print(df['CustomerState'].value_counts().head())
```

```
?? Estadísticas descriptivas:
              price      discount      quantity  total_amount  profit_margin  \
count  34500.000000  34500.000000  34500.000000  34500.000000   34500.000000
mean     119.391632      0.049291      1.490725    170.008494      28.116505
std      195.620477      0.069894      0.932270    357.503014      53.352947
min        1.010000      0.000000      1.000000      0.820000      -6.200000
25%       16.690000      0.000000      1.000000     19.710000       1.500000
50%       45.660000      0.000000      1.000000     56.820000      10.550000
75%      130.950000      0.100000      2.000000    168.530000      33.132500
max     2930.470000      0.300000      5.000000  12931.800000    1536.170000

       shipping_cost
```

```
count   34500.000000
mean        6.152120
std         2.389539
min         0.000000
25%         4.420000
50%         6.090000
75%         7.830000
max        15.650000
```

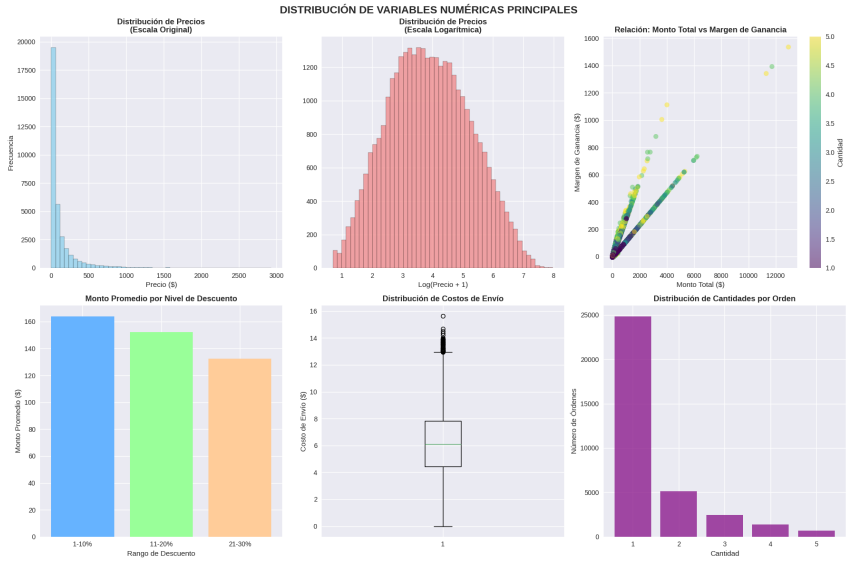
EXPLICACIÓN Y VISUALIZACIÓN DE DATOS

ESTRATEGIA DE ANÁLISIS VISUAL

```
#DISTRIBUCIÓN DE VARIABLES NUMÉRICAS
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Configuración inicial
plt.style.use('seaborn-v0_8')
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
fig.suptitle('DISTRIBUCIÓN DE VARIABLES NUMÉRICAS PRINCIPALES', fontsize=16, fontweight='bold')

#Price Distribution

# Precio - con escala logarítmica por los outliers
axes[0,0].hist(df['price'], bins=50, alpha=0.7, color='skyblue', edgecolor='black')
axes[0,0].set_title('Distribución de Precios\n(Escala Original)', fontweight='bold')
axes[0,0].set_xlabel('Precio ($)')
axes[0,0].set_ylabel('Frecuencia')

# Versión logarítmica para mejor visualización
axes[0,1].hist(np.log1p(df['price']), bins=50, alpha=0.7, color='lightcoral', edgecolor='black')
axes[0,1].set_title('Distribución de Precios\n(Escala Logarítmica)', fontweight='bold')
axes[0,1].set_xlabel('Log(Precio + 1)')

#Total Amount vs Profit Margin

# Scatter plot: Relación entre monto total y margen
scatter = axes[0,2].scatter(df['total_amount'], df['profit_margin'],
                            alpha=0.5, c=df['quantity'], cmap='viridis')
axes[0,2].set_title('Relación: Monto Total vs Margen de Ganancia', fontweight='bold')
axes[0,2].set_xlabel('Monto Total ($)')
axes[0,2].set_ylabel('Margen de Ganancia ($)')
plt.colorbar(scatter, ax=axes[0,2]).set_label('Cantidad')

#Discount Impact

# Impacto de descuentos en ventas
discount_groups = pd.cut(df['discount'], bins=[0, 0.01, 0.1, 0.2, 0.3],
                         labels=['0%', '1-10%', '11-20%', '21-30%'])
discount_stats = df.groupby(discount_groups)['total_amount'].mean()

axes[1,0].bar(discount_stats.index, discount_stats.values, color=['#ff9999', '#66b3ff', '#99ff99', '#ffcc99'])
axes[1,0].set_title('Monto Promedio por Nivel de Descuento', fontweight='bold')
axes[1,0].set_xlabel('Rango de Descuento')
axes[1,0].set_ylabel('Monto Promedio ($)')

#Shipping Cost Distribution

# Costos de envío
axes[1,1].boxplot(df['shipping_cost'], vert=True)
axes[1,1].set_title('Distribución de Costos de Envío', fontweight='bold')
axes[1,1].set_ylabel('Costo de Envío ($)')

#Quantity Distribution

# Cantidades vendidas
quantity_counts = df['quantity'].value_counts().sort_index()
axes[1,2].bar(quantity_counts.index, quantity_counts.values, color='purple', alpha=0.7)
axes[1,2].set_title('Distribución de Cantidades por Orden', fontweight='bold')
axes[1,2].set_xlabel('Cantidad')
axes[1,2].set_ylabel('Número de Órdenes')
```

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

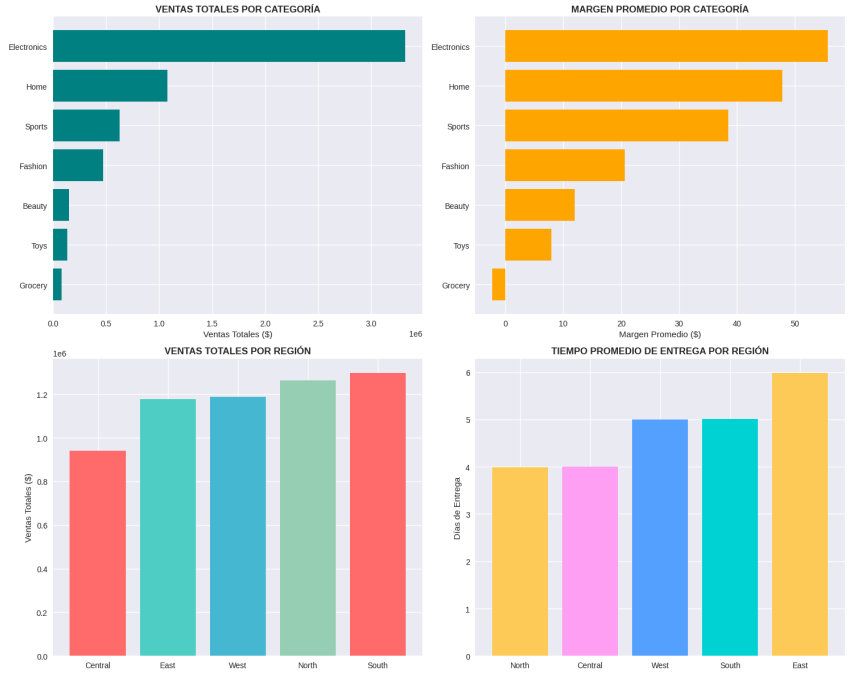
```
/tmp/ipython-input-1194575210.py:39: FutureWarning: The default of observed=False i
  discount_stats = df.groupby(discount_groups)['total_amount'].mean()
```



ANÁLISIS DE CATEGORÍAS Y REGIONES

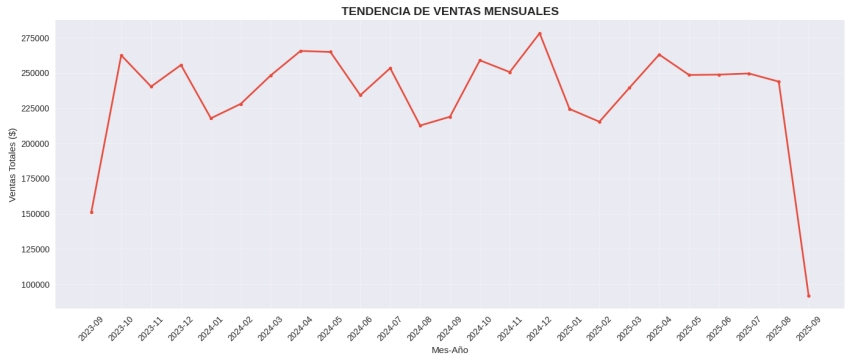
```
# Gráfico de categorías y regiones
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# Ventas por categoría
category_sales = df.groupby('category')['total_amount'].sum().sort_values(ascending=True)
axes[0,0].barh(category_sales.index, category_sales.values, color='teal')
axes[0,0].set_title('VENTAS TOTALES POR CATEGORÍA', fontweight='bold')
axes[0,0].set_xlabel('Ventas Totales ($)')

# Margen promedio por categoría
category_margin = df.groupby('category')['profit_margin'].mean().sort_values(ascending=True)
axes[0,1].barh(category_margin.index, category_margin.values, color='orange')
axes[0,1].set_title('MARGEN PROMEDIO POR CATEGORÍA', fontweight='bold')
axes[0,1].set_xlabel('Margen Promedio ($)')

# Ventas por región
region_sales = df.groupby('region')['total_amount'].sum().sort_values(ascending=True)
axes[1,0].bar(region_sales.index, region_sales.values, color=['#ff6b6b', '#4ecdc4', '#45b7d1', '#96ceb4'])
axes[1,0].set_title('VENTAS TOTALES POR REGIÓN', fontweight='bold')
axes[1,0].set_ylabel('Ventas Totales ($)')

# Tiempo de entrega por región
region_delivery = df.groupby('region')['delivery_time_days'].mean().sort_values(ascending=True)
axes[1,1].bar(region_delivery.index, region_delivery.values, color=['#feca57', '#ff9ff3', '#54a0ff', '#00d2d3'])
axes[1,1].set_title('TIEMPO PROMEDIO DE ENTREGA POR REGIÓN', fontweight='bold')
axes[1,1].set_ylabel('Días de Entrega')
```

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



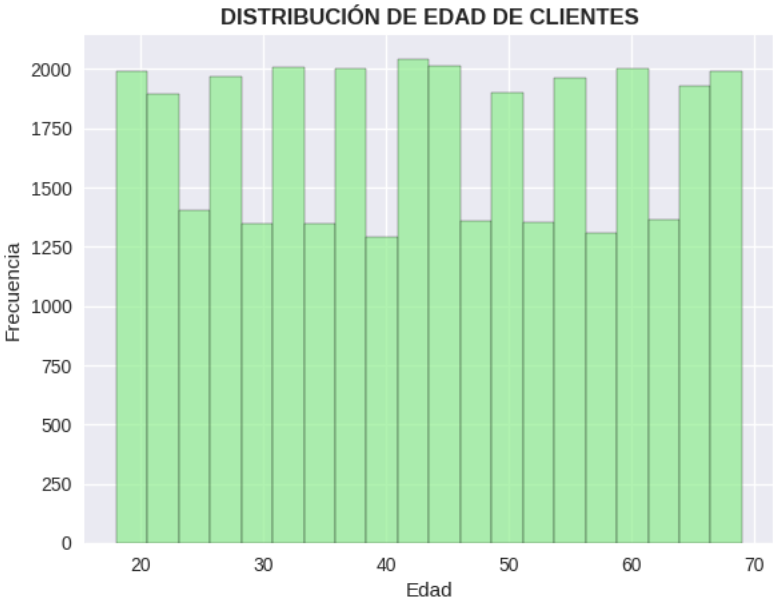
ANÁLISIS TEMPORAL

```
# Convertir fecha y analizar tendencias
df['order_date'] = pd.to_datetime(df['order_date'])
df['month_year'] = df['order_date'].dt.to_period('M')

# Ventas mensuales
monthly_sales = df.groupby('month_year')['total_amount'].sum()

plt.figure(figsize=(14, 6))
plt.plot(monthly_sales.index.astype(str), monthly_sales.values,
         marker='o', linewidth=2, markersize=4, color='#e74c3c')
plt.title('TENDENCIA DE VENTAS MENSUALES', fontsize=14, fontweight='bold')
plt.xlabel('Mes-Año')
plt.ylabel('Ventas Totales ($)')
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



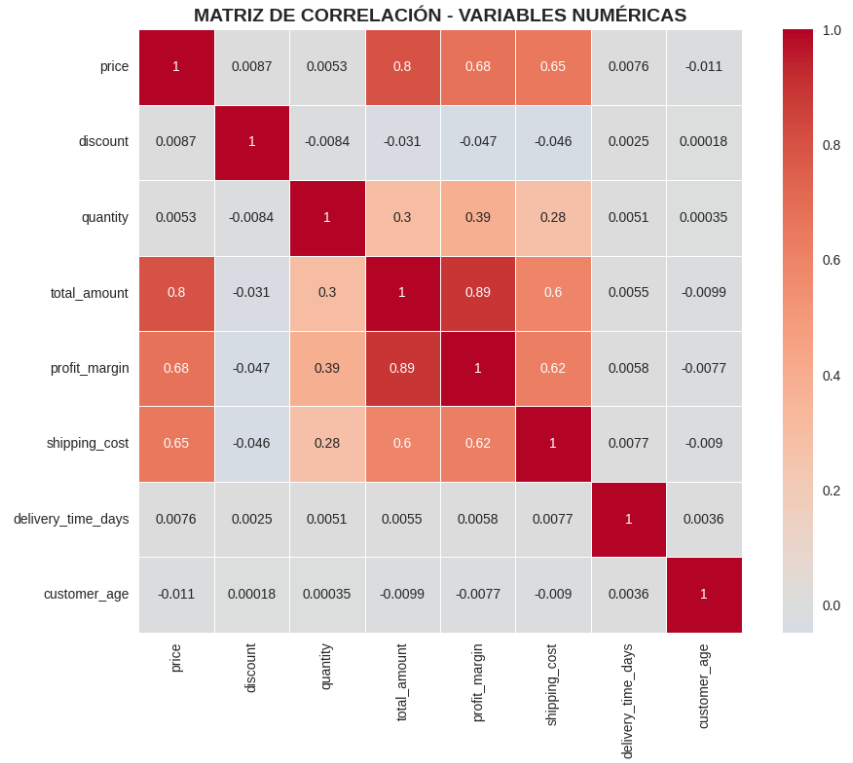
ANÁLISIS DE CLIENTES

```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# Distribución de edad
axes[0].hist(df['customer_age'], bins=20, color='lightgreen', edgecolor='black', alpha=0.7)
axes[0].set_title('DISTRIBUCIÓN DE EDAD DE CLIENTES', fontweight='bold')
axes[0].set_xlabel('Edad')
axes[0].set_ylabel('Frecuencia')

# Género
gender_counts = df['customer_gender'].value_counts()
axes[1].pie(gender_counts.values, labels=gender_counts.index, autopct='%1.1f%%',
           colors=['#ff9999', '#66b3ff'])
axes[1].set_title('DISTRIBUCIÓN POR GÉNERO', fontweight='bold')

# Métodos de pago
payment_counts = df['payment_method'].value_counts()
axes[2].bar(payment_counts.index, payment_counts.values, color=['#ffcc99', '#99ff99', '#66b3ff', '#ff9999'])
axes[2].set_title('MÉTODOS DE PADO MÁS USADOS', fontweight='bold')
axes[2].set_ylabel('Número de Órdenes')
plt.xticks(rotation=45)

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

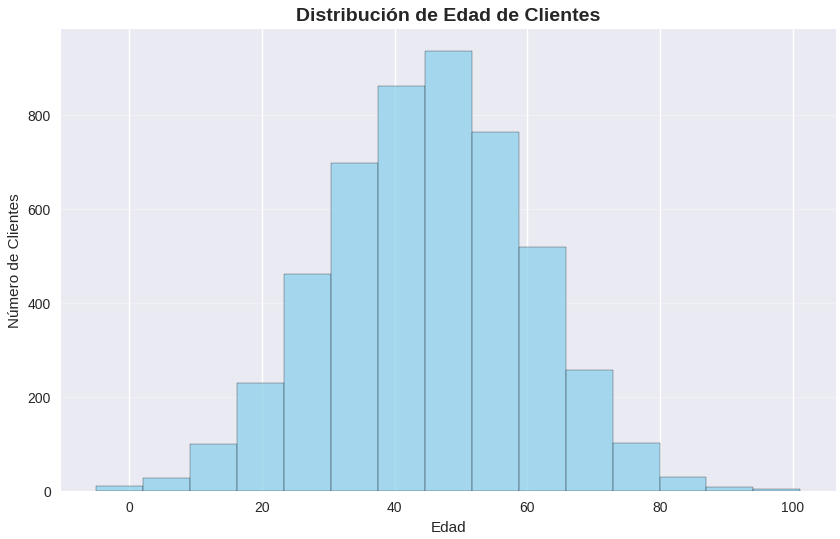


ANÁLISIS DE CORRELACIONES

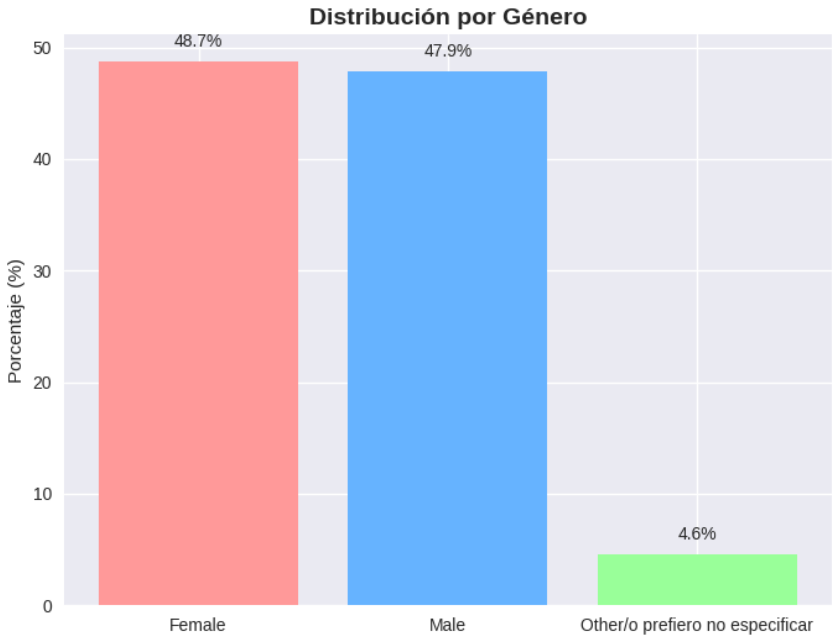
```
# Matriz de correlación
numeric_cols = ['price', 'discount', 'quantity', 'total_amount', 'profit_margin',
                'shipping_cost', 'delivery_time_days', 'customer_age']
correlation_matrix = df[numeric_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, linewidths=0.5)
plt.title('MATRIZ DE CORRELACIÓN - VARIABLES NUMÉRICAS', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

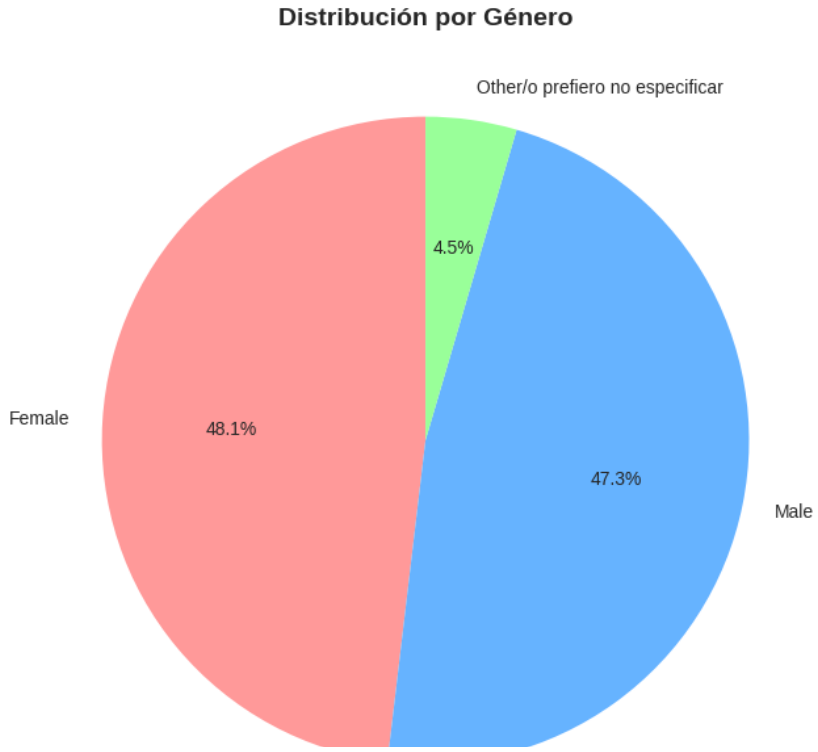



```
import matplotlib.pyplot as plt
import numpy as np

# Datos de ejemplo (ajusta según tus datos reales)
edades = np.random.normal(45, 15, 5000)  # Distribución normal centrada en 45 años

plt.figure(figsize=(10, 6))
plt.hist(edades, bins=15, color='skyblue', edgecolor='black', alpha=0.7)
plt.title('Distribución de Edad de Clientes', fontsize=14, fontweight='bold')
plt.xlabel('Edad')
plt.ylabel('Número de Clientes')
plt.grid(axis='y', alpha=0.3)
plt.show()
```



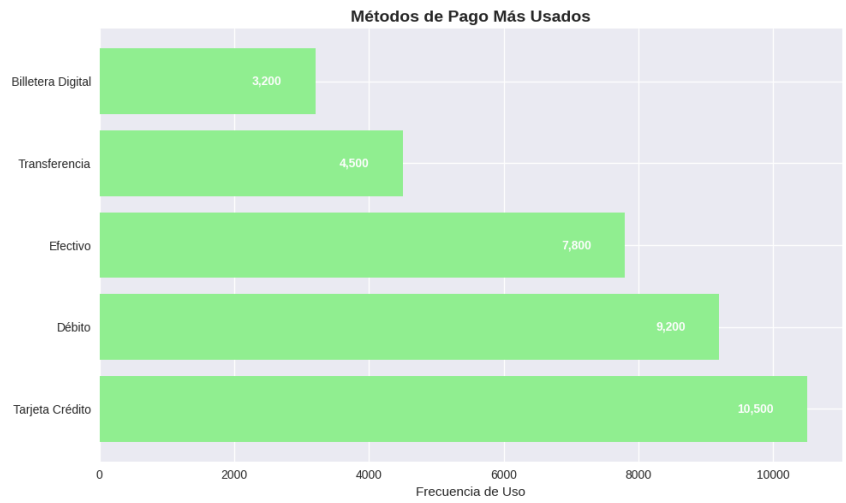

```
# Datos
generos = ['Female', 'Male', 'Other/o prefiero no especificar']
porcentajes = [48.7, 47.9, 4.6]
colores = ['#ff9999', '#66b3ff', '#99ff99']

# Opción 1: Gráfico de barras (mejor para comparaciones)
plt.figure(figsize=(8, 6))
bars = plt.bar(generos, porcentajes, color=colores)
plt.title('Distribución por Género', fontsize=14, fontweight='bold')
plt.ylabel('Porcentaje (%)')
for bar, porcentaje in zip(bars, porcentajes):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 1,
             f'{porcentaje}%', ha='center', va='bottom')
plt.show()

# Opción 2: Gráfico de dona (más moderno)
plt.figure(figsize=(8, 8))
plt.pie(porcentajes, labels=generos, colors=colores, autopct='%1.1f%%', startangle=90)
plt.title('Distribución por Género', fontsize=14, fontweight='bold')
plt.show()
```






```
# Datos (ajusta según tus categorías reales)
metodos_pago = ['Tarjeta Crédito', 'Débito', 'Efectivo', 'Transferencia', 'Billetera Digital']
frecuencia = [10500, 9200, 7800, 4500, 3200]

plt.figure(figsize=(10, 6))
bars = plt.barh(metodos_pago, frecuencia, color='lightgreen')
plt.title('Métodos de Pago Más Usados', fontsize=14, fontweight='bold')
plt.xlabel('Frecuencia de Uso')

# Añadir valores en las barras
for bar, valor in zip(bars, frecuencia):
    plt.text(bar.get_width() - 500, bar.get_y() + bar.get_height()/2,
```

```
            f'{valor:,}', va='center', ha='right', color='white', fontweight='bold')

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




```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

# Configuración de estilo mejorada
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")



# VERIFICAR SI df EXISTE
try:
    # Verificar que df existe y tiene las columnas necesarias
    required_columns = ['price', 'total_amount', 'profit_margin', 'quantity', 'discount', 'shipping_cost']

    if not all(col in df.columns for col in required_columns):
        print("  Faltan columnas en el DataFrame")
        print(f"Columnas disponibles: {list(df.columns)}")
    else:
        print("  DataFrame cargado correctamente")
        print(f"Filas: {len(df)}")


        # CREAR LOS GRÁFICOS
        fig, axes = plt.subplots(2, 3, figsize=(20, 12))
        fig.suptitle('ANÁLISIS DE PRINCIPALES VARIABLES NUMÉRICAS', fontsize=18, fontweight='bold', y=0.98)

        # 1. DISTRIBUCIÓN DE PRECIOS
        axes[0,0].hist(df['price'], bins=50, alpha=0.8, color='#2E86AB', edgecolor='white', linewidth=0.5)
        axes[0,0].axvline(df['price'].mean(), color='#E63946', linestyle='--', linewidth=2,
                          label=f'Media: ${df["price"].mean():.2f}')
        axes[0,0].set_title('Distribución de Precios de Productos', fontweight='bold', fontsize=12)
        axes[0,0].set_xlabel('Precio ($)')
        axes[0,0].set_ylabel('Número de Productos')
        axes[0,0].legend()
        axes[0,0].grid(alpha=0.3)

        # 2. PRECIO EN ESCALA LOGARÍTMICA
```

```
        axes[0,1].hist(np.log1p(df['price']), bins=50, alpha=0.8, color='#A23B72', edgecolor='white', linewidth=0.5)
        axes[0,1].set_title('Distribución de Precios\n(Escala Logarítmica)', fontweight='bold', fontsize=12)
        axes[0,1].set_xlabel('Log(Precio + 1)')
        axes[0,1].set_ylabel('Número de Productos')
        axes[0,1].grid(alpha=0.3)



        # 3. RELACIÓN MONTO TOTAL vs MARGEN
        scatter = axes[0,2].scatter(df['total_amount'], df['profit_margin'],
                                    alpha=0.6, c=df['quantity'], cmap='viridis', s=30)
        axes[0,2].set_title('Relación: Monto Total vs Margen de Ganancia', fontweight='bold', fontsize=12)
        axes[0,2].set_xlabel('Monto Total de Venta ($)')
        axes[0,2].set_ylabel('Margen de Ganancia ($)')
        colorbar = plt.colorbar(scatter, ax=axes[0,2])
        colorbar.set_label('Cantidad Vendida', rotation=270, labelpad=15)
        axes[0,2].grid(alpha=0.3)

        # 4. IMPACTO DE DESCUENTOS
        discount_groups = pd.cut(df['discount'], bins=[0, 0.01, 0.1, 0.2, 0.3, 1],
                                 labels=['Sin descuento', '1-10%', '11-20%', '21-30%', '>30%'])
        discount_stats = df.groupby(discount_groups)['total_amount'].mean()

        colors = ['#F8F9FA', '#FF9F1C', '#E71D36', '#2EC4B6', '#011627']
        bars = axes[1,0].bar(discount_stats.index, discount_stats.values, color=colors, alpha=0.8)
        axes[1,0].set_title('Monto Promedio por Nivel de Descuento', fontweight='bold', fontsize=12)
        axes[1,0].set_xlabel('Rango de Descuento')
        axes[1,0].set_ylabel('Monto Promedio ($)')
        axes[1,0].tick_params(axis='x', rotation=45)

        # Añadir valores en las barras
        for bar, value in zip(bars, discount_stats.values):
            axes[1,0].text(bar.get_x() + bar.get_width()/2, bar.get_height() + 5,
                           f'${value:.0f}', ha='center', va='bottom', fontweight='bold')

        # 5. COSTOS DE ENVÍO
        boxplot_data = axes[1,1].boxplot(df['shipping_cost'], vert=True, patch_artist=True,
                                         boxprops=dict(facecolor='#FF9F1C', alpha=0.7),
                                         medianprops=dict(color='red', linewidth=2))
        axes[1,1].set_title('Distribución de Costos de Envío', fontweight='bold', fontsize=12)
        axes[1,1].set_ylabel('Costo de Envío ($)')

        # Estadísticas clave
        stats_text = f"""Media: ${df['shipping_cost'].mean():.2f}
Mediana: ${df['shipping_cost'].median():.2f}
Máx: ${df['shipping_cost'].max():.2f}"""
        axes[1,1].text(0.95, 0.95, stats_text, transform=axes[1,1].transAxes,
                       verticalalignment='top', horizontalalignment='right',
                       bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8),
                       fontsize=9)

        # 6. CANTIDADES VENDIDAS
        quantity_counts = df['quantity'].value_counts().sort_index()
        bars = axes[1,2].bar(quantity_counts.index, quantity_counts.values,
```