# EDA Completo y Reporte Ejecutivo - Día 5

Change Status

in-progress  40 min

## Learning Objectives

1. Integrar todas las técnicas de EDA aprendidas en análisis comprehensivo
2. Crear reportes ejecutivos que comuniquen insights de manera efectiva
3. Establecer fundamentos para análisis estadísticos más avanzados



## **Ejercicio**: EDA completo y reporte ejecutivo para dataset de e-commerce

Ejercicio práctico para aplicar los conceptos aprendidos.

### Configurar dataset completo y análisis EDA:

```
import pandas as pd
import numpy as np

# Crear dataset comprehensivo de e-commerce
np.random.seed(42)
n_pedidos = 2500

# Generar fechas
fechas = pd.date_range('2023-01-01', periods=n_pedidos, freq='H')[:n_pedidos]

# Crear datos base
df = pd.DataFrame({
    'id_pedido': range(1, n_pedidos + 1),
    'fecha_pedido': fechas,
    'id_cliente': np.random.randint(1, 501, n_pedidos),
    'categoria': np.random.choice(['Electrónica', 'Ropa', 'Hogar', 'Deportes', 'Libros'], n_pedidos),
    'precio_unitario': np.round(np.random.uniform(10, 1000, n_pedidos), 2),
    'cantidad': np.random.randint(1, 5, n_pedidos),
    'metodo_pago': np.random.choice(['Tarjeta', 'PayPal', 'Efectivo', 'Transferencia'], n_pedidos, p=[0.6, 0.2, 0.15, 0.05]),
    'region': np.random.choice(['Madrid', 'Barcelona', 'Valencia', 'Sevilla', 'Bilbao'], n_pedidos),
```

```
    'tipo_cliente': np.random.choice(['Regular', 'Premium', 'VIP'], n_pedidos, p=[0.7, 0.2, 0.1])
})

# Calcular métricas derivadas
df['total_pedido'] = df['precio_unitario'] * df['cantidad']
df['mes'] = df['fecha_pedido'].dt.month
df['dia_semana'] = df['fecha_pedido'].dt.day_name()

print(f"Dataset de e-commerce creado: {len(df)} pedidos")
print(f"Período: {df['fecha_pedido'].min()} a {df['fecha_pedido'].max()}")
```

**Realizar EDA completo sistemático**:

```
# Análisis de calidad de datos
print("ANÁLISIS DE CALIDAD DE DATOS")
print("=" * 30)
print(f"Dimensiones: {df.shape}")
print(f"Tipos de datos:\n{df.dtypes}")
print(f"Valores faltantes: {df.isnull().sum().sum()}")

# Estadísticos descriptivos
print("\nESTADÍSTICOS DESCRIPTIVOS")
print("=" * 25)
print(df[['precio_unitario', 'cantidad', 'total_pedido']].describe())

# Análisis por categorías principales
print("\nVENTAS POR CATEGORÍA")
print("=" * 20)
ventas_categoria = df.groupby('categoria').agg({
    'total_pedido': ['count', 'sum', 'mean'],
    'cantidad': 'sum'
}).round(2)
print(ventas_categoria)

# Análisis temporal
print("\nVENTAS POR MES")
print("=" * 15)
ventas_mes = df.groupby('mes').agg({
    'total_pedido': 'sum',
    'id_pedido': 'count'
}).round(2)
print(ventas_mes)

# Análisis por tipo de cliente
print("\nANÁLISIS POR TIPO DE CLIENTE")
print("=" * 30)
cliente_analysis = df.groupby('tipo_cliente').agg({
    'total_pedido': ['mean', 'sum', 'count'],
```

```
    'cantidad': 'mean'
}).round(2)
print(cliente_analysis)
```

**Análisis de correlaciones y patrones**:

```
# Convertir variables categóricas para correlación
df_corr = df.copy()
df_corr['tipo_cliente_num'] = df_corr['tipo_cliente'].map({'Regular': 1, 'Premium': 2, 'VIP': 3})

# Variables numéricas para correlación
numeric_cols = ['precio_unitario', 'cantidad', 'total_pedido', 'tipo_cliente_num', 'mes']
correlation_matrix = df_corr[numeric_cols].corr()

print("\nMATRIZ DE CORRELACIÓN")
print("=" * 20)
print(correlation_matrix.round(3))

# Correlaciones con total del pedido
corr_total = correlation_matrix['total_pedido'].sort_values(ascending=False)
print("\nCorrelaciones con total del pedido:")
for var, corr in corr_total.items():
    if var != 'total_pedido':
        print(f"{var:15} | {corr:+.3f}")
```

**Detección de outliers y patrones**:

```
# Outliers en precios
Q1_precio = df['precio_unitario'].quantile(0.25)
Q3_precio = df['precio_unitario'].quantile(0.75)
IQR_precio = Q3_precio - Q1_precio

outliers_precio = df[df['precio_unitario'] > Q3_precio + 1.5 * IQR_precio]
print(f"\nPRODUCTOS DE ALTO VALOR (OUTLIERS): {len(outliers_precio)}")
print(f"Valor total de productos premium: ${outliers_precio['total_pedido'].sum():,.2f}")

# Análisis por día de la semana
ventas_dia = df.groupby('dia_semana')['total_pedido'].agg(['count', 'sum', 'mean']).round(2)
print("\nVENTAS POR DÍA DE LA SEMANA")
print("=" * 30)
print(ventas_dia.sort_values('sum', ascending=False))
```

**Crear reporte ejecutivo simplificado**:

```
# Calcular métricas clave para reporte
total_ventas = df['total_pedido'].sum()
pedidos_promedio = df['total_pedido'].mean()
categoria_top = df.groupby('categoria')['total_pedido'].sum().idxmax()
ventas_categoria_top = df.groupby('categoria')['total_pedido'].sum().max()
region_top = df.groupby('region')['total_pedido'].sum().idxmax()

# Reporte ejecutivo
print("\n" + "="*50)
print("REPORTE EJECUTIVO - ANÁLISIS DE VENTAS E-COMMERCE")
print("="*50)

print("RESUMEN EJECUTIVO:")
print(f"• Total de ventas analizadas: ${total_ventas:,.2f}")
print(f"• Pedidos promedio: ${pedidos_promedio:.2f}")
print(f"• Categoría más vendida: {categoria_top} (${ventas_categoria_top:,.2f})")
print(f"• Región con más ventas: {region_top}")

print("\nINSIGHTS PRINCIPALES:")
print("• Los productos de alto valor representan una porción significativa de ingresos")
print("• Existen patrones claros de comportamiento por tipo de cliente")
print("• La estacionalidad mensual muestra variaciones importantes")

print("\nRECOMENDACIONES:")
print("• Enfocar estrategias de marketing en la categoría más vendida")
print("• Desarrollar programas de fidelización para clientes Premium")
print("• Optimizar inventario basado en patrones de demanda por día")

print("="*50)
```

**Verificación**: Evalúa si tu análisis responde preguntas clave de negocio: ¿Qué vende mejor? ¿Quiénes son los mejores clientes? ¿Cuándo ocurren las ventas? ¿Qué patrones requieren acción inmediata?

**Requerimientos:**

Python con Pandas, NumPy
matplotlib/seaborn opcionales para visualizaciones avanzadas
Dataset completo para análisis comprehensivo