Filtrado, Agrupación y Merge de Datos - Día 4

Change Status

pending 40 min

Learning Objectives

1. Dominar técnicas avanzadas de filtrado condicional usando query() y boolean indexing
2. Aplicar operaciones de agrupación con groupby() para análisis por categorías
3. Combinar datasets de diferentes fuentes usando operaciones de merge

Ejercicio: Análisis completo combinando filtrado, agrupación y merge

Ejercicio práctico para aplicar los conceptos aprendidos.

Crear datasets relacionados:

```
import pandas as pd
import numpy as np

# Dataset de ventas
ventas = pd.DataFrame({
    'id_venta': range(1, 11),
    'id_cliente': np.random.choice([1, 2, 3, 4, 5], 10),
    'id_producto': np.random.choice([101, 102, 103, 104], 10),
    'cantidad': np.random.randint(1, 5, 10),
    'fecha': pd.date_range('2024-01-01', periods=10, freq='D')
})

# Dataset de clientes
clientes = pd.DataFrame({
    'id_cliente': [1, 2, 3, 4, 5],
    'nombre': ['Ana', 'Carlos', 'María', 'Juan', 'Luis'],
```

```
    'ciudad': ['Madrid', 'Barcelona', 'Madrid', 'Valencia', 'Sevilla']
})

# Dataset de productos
productos = pd.DataFrame({
    'id_producto': [101, 102, 103, 104],
    'nombre': ['Laptop', 'Mouse', 'Teclado', 'Monitor'],
    'precio': [1200, 25, 80, 300],
    'categoria': ['Electrónica', 'Accesorios', 'Accesorios', 'Electrónica']
})

print("Datasets creados:")
print(f"Ventas: {ventas.shape}")
print(f"Clientes: {clientes.shape}")
print(f"Productos: {productos.shape}")
```

Filtrado avanzado con query():

```
# Ventas del mes actual con query
ventas_recientes = ventas.query('fecha >= "2024-01-05"')
print(f"\nVentas recientes: {len(ventas_recientes)}")

# Productos caros usando variable externa
precio_limite = 100
productos_caros = productos.query('precio >= @precio_limite')
print(f"Productos caros (>= {precio_limite}): {productos_caros['nombre'].tolist()}")
```

Agrupación y agregación:

```
# Ventas por producto
ventas_por_producto = ventas.groupby('id_producto')['cantidad'].sum()
print(f"\nVentas por producto:\n{ventas_por_producto}")

# Estadísticas por cliente
stats_por_cliente = ventas.groupby('id_cliente').agg({
    'cantidad': ['sum', 'mean'],
    'id_venta': 'count'
})
print(f"\nEstadísticas por cliente:\n{stats_por_cliente}")
```

Merge para análisis completo:

```
# Unir ventas con productos
ventas_productos = pd.merge(ventas, productos, on='id_producto')

# Calcular totales
ventas_productos['total'] = ventas_productos['cantidad'] * ventas_productos['precio']

# Unir con clientes
analisis_completo = pd.merge(ventas_productos, clientes, on='id_cliente')

print(f"\nAnálisis completo (primeras 5 filas):\n{analisis_completo.head()}")

# Análisis por ciudad
ventas_por_ciudad = analisis_completo.groupby('ciudad')['total'].sum()
print(f"\nVentas totales por ciudad:\n{ventas_por_ciudad}")
```

Filtrado final:

```
# Clientes con compras > 1000
clientes_top = analisis_completo.groupby(['id_cliente', 'nombre'])['total'].sum()
clientes_top = clientes_top[clientes_top > 1000]
print(f"\nClientes con compras > 1000:\n{clientes_top}")
```

Verificación: Examina cada resultado intermedio para entender cómo cada operación (filtrado, agrupación, merge) contribuye al análisis final.

Requerimientos:

Python con Pandas instalado
NumPy disponible
Datasets relacionados para practicar joins
Conocimiento de DataFrames básicos (del día 2)