Clase 032 — Pandas: eval y query

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3 § 3.13 High-Performance Pandas: eval and query. · Duración estimada: 45 min.

Clase 032 — Pandas: eval y query

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3 § 3.13 High-Performance Pandas: eval and query. · Duración estimada: 45 min.*

Objetivo

Que el alumno conozca df.eval y df.query — herramientas para expresar operaciones y filtros con sintaxis tipo SQL en strings. Útiles para legibilidad en cadenas largas y, en datasets muy grandes, también más rápidos (usan numexpr).

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Filtrar con df.query("col > 10 and other == 'X'").
2. Calcular columnas nuevas con df.eval('z = x + y') o df.eval('x * 2').
3. Referenciar variables locales en query/eval con prefijo @: df.query('x > @threshold').
4. Decidir cuándo usar query (legibilidad en cadenas largas) vs filtro tradicional (mejor autocompletado IDE).
5. Saber que el speedup real solo aparece con datasets >10k filas y expresiones complejas.

Temas

#	Tema	Por qué importa
1	df.query — sintaxis tipo SQL	Una sola string en vez de máscara compuest
2	df.eval — expresiones aritméticas	Calcula columnas sin temporales.
3	Variables locales con @	Pasar valores del scope.
4	numexpr para speedup	Solo en datasets grandes.
5	Trade-off: legibilidad vs introspección ID	Query strings no tienen autocompletado.

Definiciones y características

df.query()

: Filtro como string tipo SQL: df.query('precio > 100 and categoria == "A"'). Más legible que máscara booleana compuesta cuando hay >2 condiciones.

df.eval()

: Evalúa expresiones aritméticas: df.eval('total = precio * cantidad'). Con inplace=True añade la columna al DF.

Variables locales (@var)

: Dentro de query/eval, prefijo @ referencia variables del scope Python: df.query('x > @threshold').

numexpr

: Motor de cómputo que vectoriza expresiones en C/SIMD. Usado por eval/query bajo el capó cuando los

datasets son grandes. Acelera operaciones aritméticas complejas.

Trade-off legibilidad vs IDE

: Query strings: más legibles para humanos. Filtros tradicionales: autocomplete del IDE, type checking. Elige según contexto.

Dataset / recursos

Sintético: DataFrame grande para benchmark. Sin descarga.

Ejercicios

1. Filter tradicional vs query. df[(df.a > 10) & (df.b < 5) & (df.c == 'x')] vs df.query('a > 10 and b < 5 and c == "x"'). Compara legibilidad.

2. Variable local. threshold = 100; filtra con df.query('precio > @threshold').

3. eval para nueva columna. df.eval('total = precio * cantidad', inplace=True).

4. Benchmark. Genera df 1M filas. Compara filter tradicional vs query con %timeit.

5. eval con inplace=False vs cálculo tradicional df['total'] = df['precio'] * df['cantidad'] — verifica resultados idénticos.

Homework verificable

Notebook con df 100k filas: (a) 3 filtros equivalentes (mask, query, query con @var); (b) eval para crear 2 columnas derivadas; (c) benchmark tradicional vs query en N=100k y N=1M; (d) reporte: cuándo conviene cada uno.

Criterio de aceptación: Resultados idénticos entre métodos. Speedup de query aparece en N≥100k.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
UndefinedVariableError: name 'X' is not de	Variable Python no prefijada con @. Fix: d
Strings dentro de query con comillas doble	Mezcla de quotes. Fix: usa quotes opuestas
df.eval('col_nueva = ...') no añade la col	Sin inplace=True. Fix: df.eval('col = xy',
query/eval más lento que máscara tradicion	Para datasets pequeños (<10k filas), el ov
Funciones custom no funcionan en query	Solo aritmética + operadores. Fix: df.quer

Preguntas frecuentes

¿query o máscara tradicional?

Máscara para filtros simples (1-2 condiciones) y autocomplete del IDE. query para filtros largos (4+ condiciones), parametrizables (@var), o cuando el lector lo encuentra más claro.

¿eval y query siempre son más rápidos?

No — solo en datasets grandes (>100k) con expresiones complejas. Para pequeños son iguales o ligeramente más lentos.

¿Qué operadores acepta query?

Aritméticos (+ - / *), comparación (==, <, >, <=, >=, !=), boolean (and, or, not o &, |, ~), in, not in. No funciones.

¿query reemplaza a SQL en pandas?

Para filtros, sí (mismo nivel expresivo). Para joins y aggregations, no — usa merge y groupby clásicos. Para datasets >1GB, considera DuckDB directo.

¿Hay eval peligroso (security)?

pd.eval no usa exec() Python — es parser propio limitado. No ejecuta arbitrary code. Pero: nunca pases strings de usuario sin sanitizar a query/eval.

Referencias

- VanderPlas, cap. 3 § 3.13.
- pandas eval/query docs
- numexpr project

Siguiente clase

Clase 033 — Polars: DataFrames modernos

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import numpy as np
import pandas as pd
import time
rng = np.random.default_rng(42)
```

Archivos complementarios

- notebook.ipynb