Understand Polars Expressions when you're used to pandas

Quansight Labs

Marco Gorelli (Senior Software Engineer at Quansight Labs, pandas, Polars, Narwhals, and more)

At Data Umbrella, 19 November, 2024!

Dutch expressions

Quansight Labs

Dutch expressions

Quansight Labs

“Too bad. Peanut butter”

Quansight Labs

Why care about Polars
Crash course

Can we reimplement expressions using pandas?

Awkward silence

5 mins → 13 mins → 7 mins → *x* seconds → 5 minutes - *x* seconds

Expressions:
- An introduction
- Multi-input expressions
- Multi-output expressions
- Group-by

Engaging Q&A

Quansight Labs

Why care

(about Polars)

?

Why care (about Polars)?

Quansight Labs

What Makes Polars So Special for G-Research?

Speed That Speaks Volumes

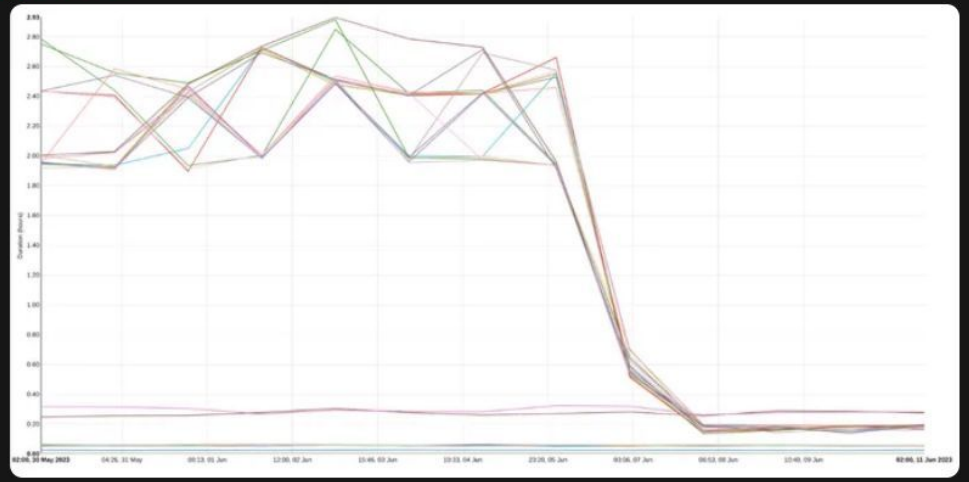
Teams at G-Research are genuinely excited about how much quicker Polars is, especially when dealing with gargantuan group bys, complex strings or juxtaposed joins that could slow down workflows to a crawl. One user pointed out he helped a colleague to get a 150x speedup over our previous implementation. That's not just an improvement for G-Research; it's a game-changer.

Why care (about Polars)?

Quansight Labs

Sky | Showmax @ShowmaxDevs Jun 19

We are eager to offer film or series recommendations instantly and we made another step towards blazing-fast data processing. We are using Polars now, a modern data framework written in #Rust. On several tasks we recorded ten times shorter #DataProcessing times.
#SVOD



Jun 19, 2023 · 9:22 AM UTC

What M
Researc

Speed Tha

Teams at G-Re
especially whe
juxtaposed joi
he helped a co
That's not just

out
n.

Why care (about Polars)?

Quansight Labs

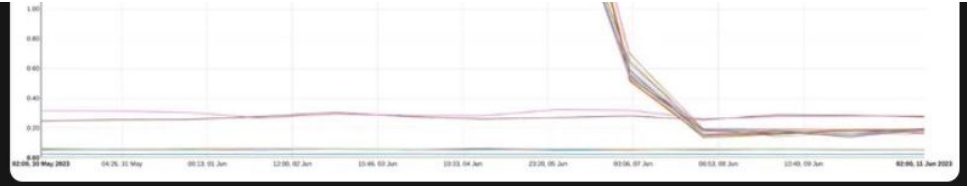
Sky | Showmax @ShowmaxDevs Jun 19

We are eager to offer film or series recommendations instantly and we made another step towards blazing-fast data processing.

Check Technologies saves 25% of cloud expenses with Polars

Wed, 6 Mar 2024

teams at G-Re
especially whe
juxtaposed joi
he helped a co
That's not just

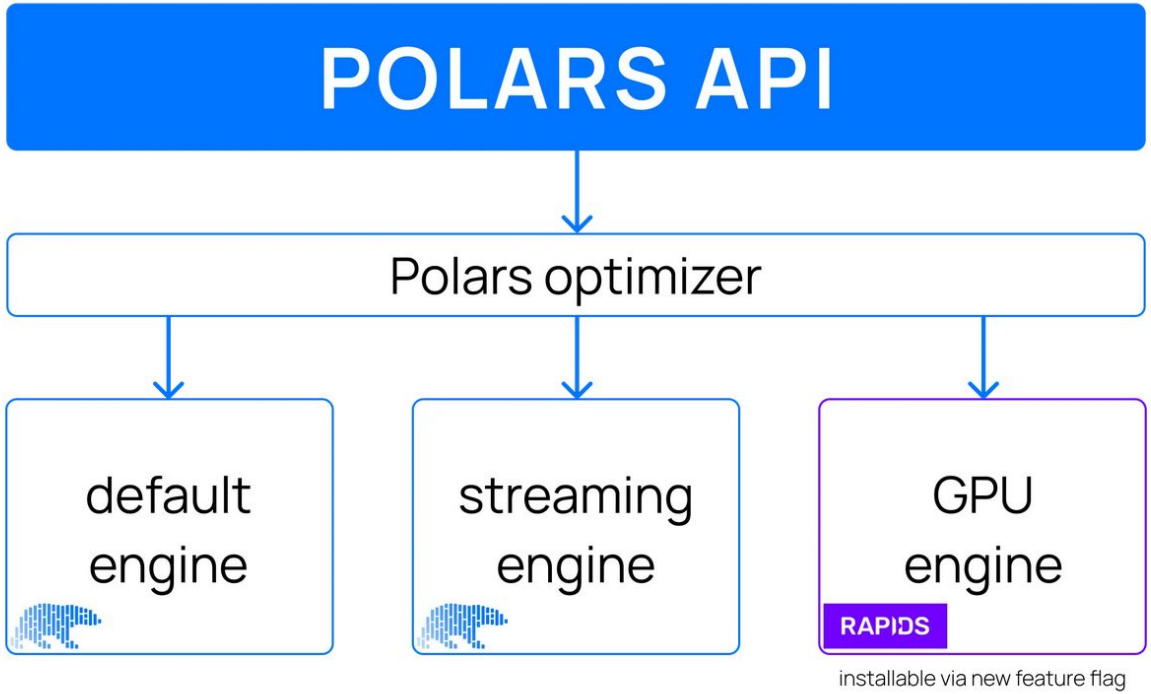


Jun 19, 2023 · 9:22 AM UTC

out
n.

Why care (about Polars)?

Quansight Labs



C es 25%

o olars

Wed

juxtaposed joi out

he helped a co n.

That's not just

Quansight Labs

Polars crash course

Polars crash course: DataFrame

Quansight Labs

The most fundamental data structure: `DataFrame`

- A collection of `Series`, all of the same length
- Each `Series`' elements have the same type (e.g. `str`, `int64`, `boolean`, `float64`, `date`, ...)
- All data types support missing values

```python
import polars as pl

assets = pl.read_parquet('assets.parquet')
```

```python
assets
```

shape: (24_564, 3)

symbol	date	price
str	date	f64
"ABBV"	2022-01-31	131.87
"ABBV"	2022-02-01	131.976
"ABBV"	2022-02-02	133.537
"ABBV"	2022-02-03	135.57
"ABBV"	2022-02-04	135.492
...	...	...
"XOM"	2023-01-17	112.93
"XOM"	2023-01-18	110.61
"XOM"	2023-01-19	111.32
"XOM"	2023-01-20	113.35
"XOM"	2023-01-23	112.76

Polars crash course: Series

Quansight Labs

You can get a 1D Series out of a DataFrame. E.g. to get column `'price'`:

- `DataFrame['price']`

```
1  assets['price']
```

shape: (24_564,)

price
f64
131.87
131.976
133.537
135.57
135.492
...
112.93
110.61
111.32
113.35
112.76

Polars crash course: Series

Quansight Labs

You can operate on that Series, and insert it back into the dataframe using `DataFrame.with_columns`

```
assets.with_columns(
    price_doubled = assets['price'] * 2
)
```

shape: (24_564, 4)

symbol	date	price	price_doubled
str	date	f64	f64
"ABBV"	2022-01-31	131.87	263.74
"ABBV"	2022-02-01	131.976	263.952
"ABBV"	2022-02-02	133.537	267.074
"ABBV"	2022-02-03	135.57	271.14
"ABBV"	2022-02-04	135.492	270.984
…	…	…	…
"XOM"	2023-01-17	112.93	225.86
"XOM"	2023-01-18	110.61	221.22
"XOM"	2023-01-19	111.32	222.64
"XOM"	2023-01-20	113.35	226.7
"XOM"	2023-01-23	112.76	225.52

Polars crash course: Series

Quansight Labs

You can operate on that Series, and insert it back into the dataframe using `DataFrame.with_columns`

🛑 But WAIT!

...should you?

```
assets.with_columns(
    price_doubled = assets['price'] * 2
)
```

shape: (24_564, 4)

symbol	date	price	price_doubled
str	date	f64	f64
"ABBV"	2022-01-31	131.87	263.74
"ABBV"	2022-02-01	131.976	263.952
"ABBV"	2022-02-02	133.537	267.074
"ABBV"	2022-02-03	135.57	271.14
"ABBV"	2022-02-04	135.492	270.984
...	...	...	...
"XOM"	2023-01-17	112.93	225.86
"XOM"	2023-01-18	110.61	221.22
"XOM"	2023-01-19	111.32	222.64
"XOM"	2023-01-20	113.35	226.7
"XOM"	2023-01-23	112.76	225.52

Polars crash course: Expressions

Quansight Labs

“Came for the speed, stayed for the syntax” every Polars user ever

Quansight Labs

Expressions

(a light introduction)

Expressions: selection

Quansight Labs

Polars newbies are often quick to gain an intuition for how expressions behave

But what exactly does `pl.col('price')*2` mean?

```
assets.with_columns(
    price_doubled = pl.col('price') * 2
)
```

shape: (24_564, 4)

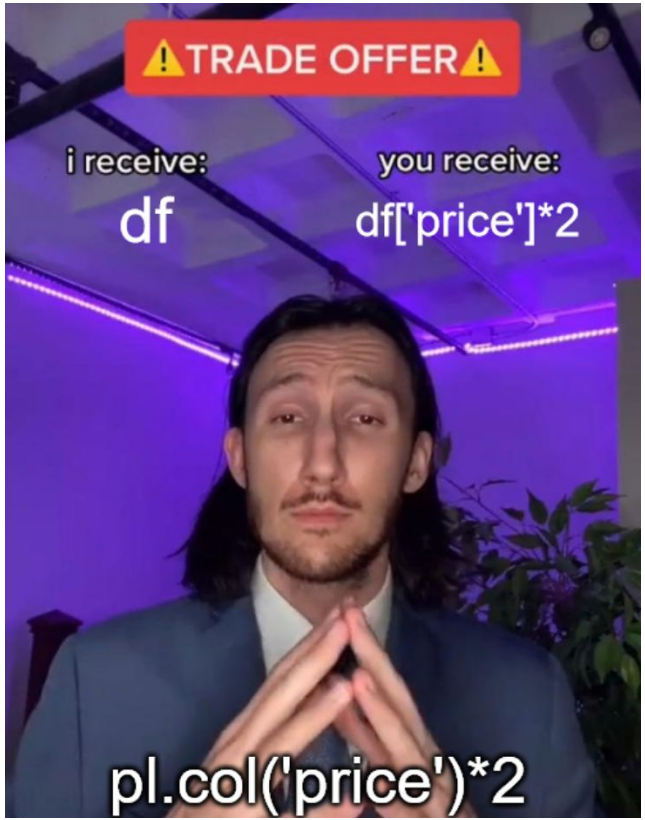
symbol	date	price	price_doubled
str	date	f64	f64
"ABBV"	2022-01-31	131.87	263.74
"ABBV"	2022-02-01	131.976	263.952
"ABBV"	2022-02-02	133.537	267.074
"ABBV"	2022-02-03	135.57	271.14
"ABBV"	2022-02-04	135.492	270.984
...	...	...	...
"XOM"	2023-01-17	112.93	225.86
"XOM"	2023-01-18	110.61	221.22
"XOM"	2023-01-19	111.32	222.64
"XOM"	2023-01-20	113.35	226.7
"XOM"	2023-01-23	112.76	225.52

Expressions: a step back

Quansight Labs

`pl.col('price')*2`:
"given some dataframe `df`, return `df['price']*2`"

So, what is an expression?



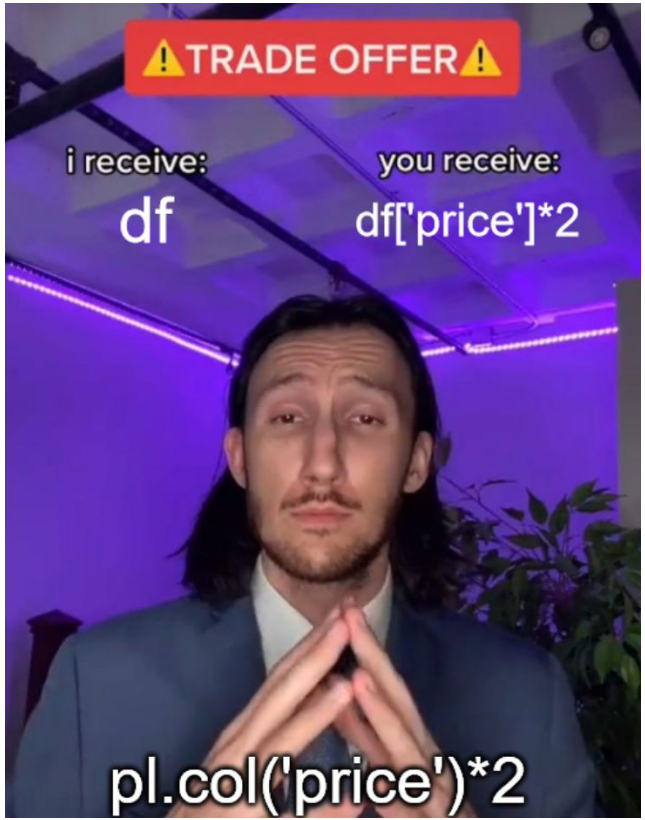
Expressions: a step back

Quansight Labs

`pl.col('price')*2`:
"given some dataframe `df`, return `df['price']*2`"

So, what is an expression?

An expression is a function from a Dataframe to a Series



Functions: a detour

Quansight Labs

Here's a function:

```
lambda x: x*2
```

By itself, it doesn't produce a value. It only produces a value when you pass it an input

Functions: a detour

Quansight Labs

Here's a function:

```
lambda x: x*2
```

By itself, it doesn't produce a value. It only produces a value when you pass it an input

```
1  (lambda x: x*2)(3)
```

```
6
```

```
1  (lambda x: x*2)(4)
```

```
8
```

Expressions are...functions

Quansight Labs

```
(lambda df: df['price']*2)(assets)
```

shape: (24_564,)

price
f64
263.74
263.952
267.074
271.14
270.984
...
225.86
221.22
222.64
226.7
225.52

```
assets.select(pl.col('price')*2)
```

shape: (24_564, 1)

price
f64
263.74
263.952
267.074
271.14
270.984
...
225.86
221.22
222.64
226.7
225.52

Expressions are...functions

Quansight Labs

An expression is a “recipe” for creating a Series.

Polars doesn’t need to evaluate it right away ⇒ optimisations can happen!

```
.with_columns(
    lagged_mean_24h=pl.col("price").shift(1).rolling_mean(24),
    lagged_max_24h=pl.col("price").shift(1).rolling_max(24),
    lagged_min_24h=pl.col("price").shift(1).rolling_min(24),
    lagged_mean_7d=pl.col("price").shift(1).rolling_mean(7 * 24)
    lagged_max_7d=pl.col("price").shift(1).rolling_max(7 * 24),
    lagged_min_7d=pl.col("price").shift(1).rolling_min(7 * 24),
)
```

Expressions are…functions

Quansight Labs

An expression is a “recipe” for creating a Series.

Polars doesn’t need to evaluate it right away ⇒ optimisations can happen!

pl.col(‘price’).shift(1) is shared between expressions, and can just be calculated once and reused!

```
.with_columns(
    lagged_mean_24h=pl.col("price").shift(1).rolling_mean(24),
    lagged_max_24h=pl.col("price").shift(1).rolling_max(24),
    lagged_min_24h=pl.col("price").shift(1).rolling_min(24),
    lagged_mean_7d=pl.col("price").shift(1).rolling_mean(7 * 24)
    lagged_max_7d=pl.col("price").shift(1).rolling_max(7 * 24),
    lagged_min_7d=pl.col("price").shift(1).rolling_min(7 * 24),
)
```

Quansight Labs

Expressions

(multiple inputs)

Expressions with multiple inputs

Quansight Labs

Q: “For each date, what was the total price across all assets?”

```
assets_wide
```

shape: (246, 4)

date	ABBV	EWC	XLP
date	f64	f64	f64
2022-01-31	131.87	37.366	74.058
2022-02-01	131.976	37.845	73.99
2022-02-02	133.537	38.002	74.877
2022-02-03	135.57	37.356	74.897
2022-02-04	135.492	37.561	73.98
...	...	...	...
2023-01-17	152.83	34.95	74.79
2023-01-18	149.2	34.56	72.75
2023-01-19	148.71	34.62	72.06
2023-01-20	149.59	35.1	72.62
2023-01-23	148.55	35.33	72.85

Expressions with multiple inputs

Quansight Labs

Q: “For each date, what was the total price across all assets?”

A: “use `sum_horizontal`!”

```
assets_wide
```

shape: (246, 4)

date	ABBV	EWC	XLP
date	f64	f64	f64
2022-01-31	131.87	37.366	74.058
2022-02-01	131.976	37.845	73.99
2022-02-02	133.537	38.002	74.877
2022-02-03	135.57	37.356	74.897
2022-02-04	135.492	37.561	73.98
…	…	…	…
2023-01-17	152.83	34.95	74.79
2023-01-18	149.2	34.56	72.75
2023-01-19	148.71	34.62	72.06
2023-01-20	149.59	35.1	72.62
2023-01-23	148.55	35.33	72.85

Expressions with multiple inputs

Quansight Labs

"An expression is a function from a DataFrame to a Series"

Multiple columns can be selected from the input Series

```
# (lambda df: df['ABBV'] + df['EWC'] + df['XLP'])(assets_wide)  # kinda
assets_wide.with_columns(total = pl.sum_horizontal('ABBV', 'EWC', 'XLP'))
```

shape: (246, 5)

date	ABBV	EWC	XLP	total
date	f64	f64	f64	f64
2022-01-31	131.87	37.366	74.058	243.294
2022-02-01	131.976	37.845	73.99	243.811
2022-02-02	133.537	38.002	74.877	246.416
2022-02-03	135.57	37.356	74.897	247.823
2022-02-04	135.492	37.561	73.98	247.033
...	...	...	...	...
2023-01-17	152.83	34.95	74.79	262.57
2023-01-18	149.2	34.56	72.75	256.51
2023-01-19	148.71	34.62	72.06	255.39
2023-01-20	149.59	35.1	72.62	257.31
2023-01-23	148.55	35.33	72.85	256.73

Expressions with multiple inputs

Quansight Labs

"An expression is a function from a DataFrame to a Series"

Multiple columns can be selected from the input Series...

...but the output is always just a single Series

```
# (lambda df: df['ABBV'] + df['EWC'] + df['XLP'])(assets_wide)  # kinda
assets_wide.with_columns(total = pl.sum_horizontal('ABBV', 'EWC', 'XLP'))
```

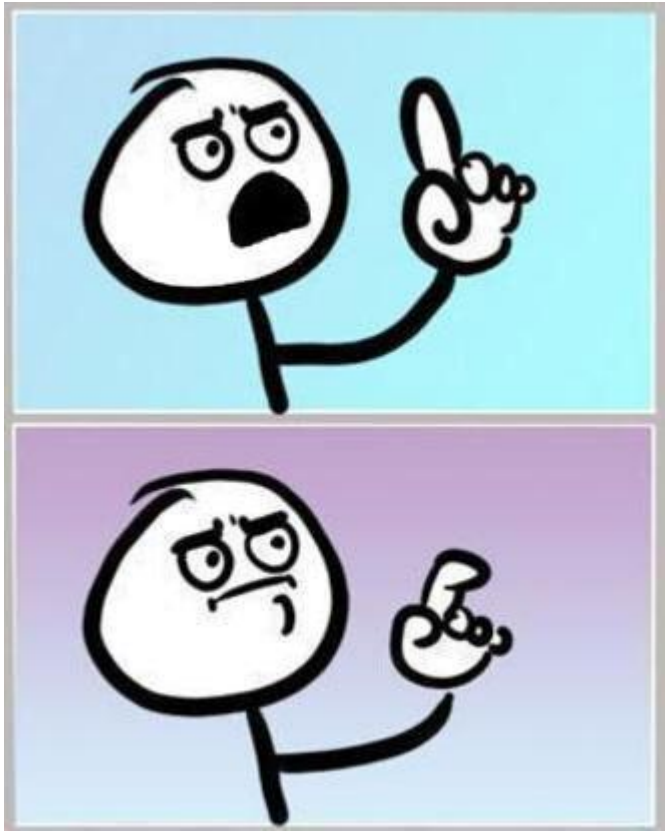
shape: (246, 5)

date	ABBV	EWC	XLP	total
date	f64	f64	f64	f64
2022-01-31	131.87	37.366	74.058	243.294
2022-02-01	131.976	37.845	73.99	243.811
2022-02-02	133.537	38.002	74.877	246.416
2022-02-03	135.57	37.356	74.897	247.823
2022-02-04	135.492	37.561	73.98	247.033
...	...	...	...	...
2023-01-17	152.83	34.95	74.79	262.57
2023-01-18	149.2	34.56	72.75	256.51
2023-01-19	148.71	34.62	72.06	255.39
2023-01-20	149.59	35.1	72.62	257.31
2023-01-23	148.55	35.33	72.85	256.73

Quansight Labs

BUT WAIT

What about pl.col(‘a’, ‘b’, ‘c’)?



Quansight Labs

Expressions

(they still makes sense, I promise)

Expressions with multiple…outputs?

Quansight Labs

"An expression is a function from a dataframe to a Series"

Then why does `pl.col('a', 'b', 'c')*2` produce 3 Series?

```
1 df
```

shape: (10, 3)

a	b	c
i64	i64	i64
4	8	8
5	4	7
7	2	8
9	8	5
0	2	8
1	4	3
8	6	4
9	5	7
2	0	1
3	0	3

```
1 df.select(pl.col('a', 'b', 'c')*2)
```

shape: (10, 3)

a	b	c
i64	i64	i64
8	16	16
10	8	14
14	4	16
18	16	10
0	4	16
2	8	6
16	12	8
18	10	14
4	0	2
6	0	6

Expressions with multiple...outputs?

Quansight Labs

"An expression is a function from a dataframe to a Series"

Then why does `pl.col('a', 'b', 'c')*2` produce 3 Series?

wut?

```
1  df
```

shape: (10, 3)

a	b	c
i64	i64	i64
4	8	8
5	4	7
7	2	8
9	8	5
0	2	8
1	4	3
8	6	4
9	5	7
2	0	1
3	0	3

```
1  df.select(pl.col('a', 'b', 'c')*2)
```

shape: (10, 3)

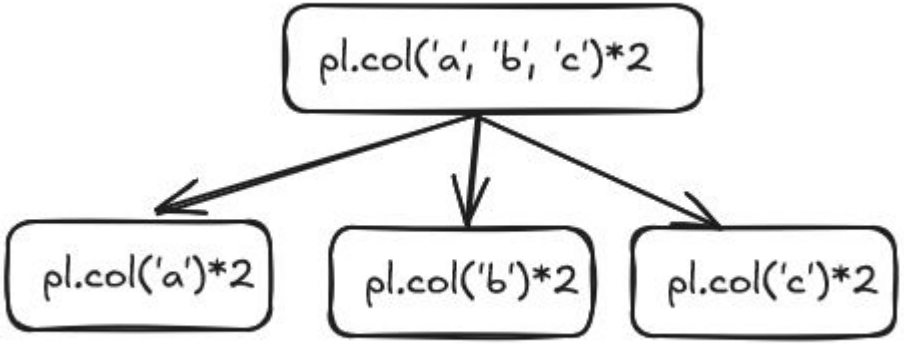
a	b	c
i64	i64	i64
8	16	16
10	8	14
14	4	16
18	16	10
0	4	16
2	8	6
16	12	8
18	10	14
4	0	2
6	0	6

Expressions with multiple...outputs?

Quansight Labs

`pl.col('a', 'b', 'c')*2` gets expanded out into three expressions

That's how it gets to produce three Series as output ;)



Expressions with multiple...outputs?

Quansight Labs

"An expression is a function from a dataframe to a Series"

Then why does `pl.col('a', 'b', 'c')*2` produce 3 Series?

Ok, all clear!

```
1 df
```

shape: (10, 3)

a	b	c
i64	i64	i64
4	8	8
5	4	7
7	2	8
9	8	5
0	2	8
1	4	3
8	6	4
9	5	7
2	0	1
3	0	3

```
1 df.select(pl.col('a', 'b', 'c')*2)
```

shape: (10, 3)

a	b	c
i64	i64	i64
8	16	16
10	8	14
14	4	16
18	16	10
0	4	16
2	8	6
16	12	8
18	10	14
4	0	2
6	0	6

Expressions: summary

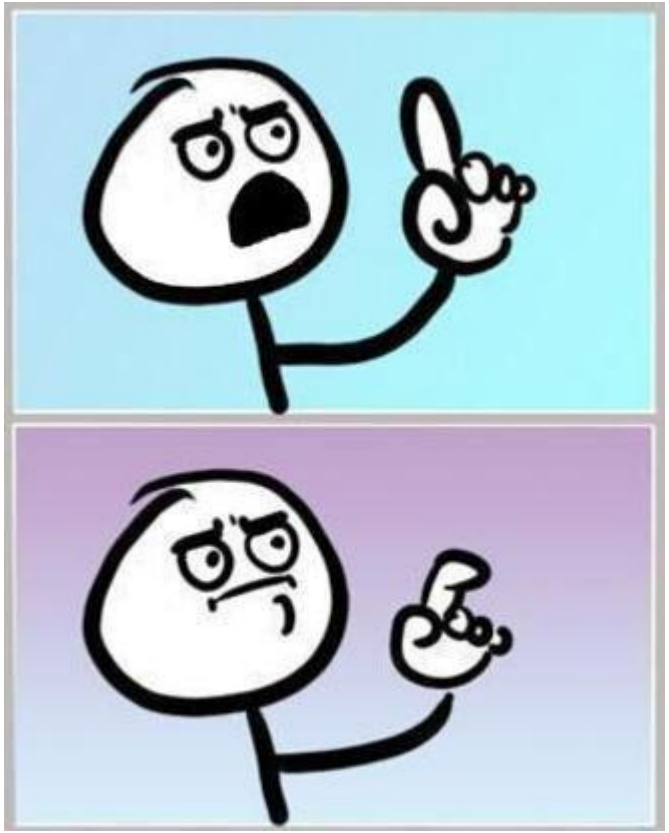
Quansight Labs

- An expression is a function from a DataFrame to a Series
- The expression can use multiple columns from the input dataframe, but it only returns a single Series
- pl.col('a', 'b', 'c') is shorthand for 3 expressions

Quansight Labs

BUT WAIT

What about group-by aggregations?



Expressions in group-by

Quansight Labs

We can pass an expression to `agg`

We then get a horizontal Series for each group

```
animals
```

shape: (7, 2)

name	weight
str	f64
"Puffin"	0.32
"Puffin"	0.35
"Puffin"	0.45
"Dolphin"	3010.0
"Dolphin"	3560.0
"Manatee"	490.0
"Manatee"	511.0

```
animals.group_by('name').agg(pl.col('weight'))
```

shape: (3, 2)

name	weight
str	list[f64]
"Puffin"	[0.32, 0.35, 0.45]
"Manatee"	[490.0, 511.0]
"Dolphin"	[3010.0, 3560.0]

Expressions in group-by

Quansight Labs

We can pass an expression to `agg`

We then get a horizontal Series for each group

```
animals
```

shape: (7, 2)

name	weight
str	f64
"Puffin"	0.32
"Puffin"	0.35
"Puffin"	0.45
"Dolphin"	3010.0
"Dolphin"	3560.0
"Manatee"	490.0
"Manatee"	511.0

```
animals.group_by('name').agg(pl.col('weight'))
```

shape: (3, 2)

name	weight
str	list[f64]
"Puffin"	[0.32, 0.35, 0.45]
"Manatee"	[490.0, 511.0]
"Dolphin"	[3010.0, 3560.0]

Expressions in group-by: a step back

Quansight Labs

What happens if we do `pl.col('weight').sum()`?

- pl.col('weight'): f64
- pl.col('weight').sum(): f64

The data type is preserved

```
1 puffin.select(pl.col('weight'))
```

shape: (3, 1)

weight
f64
0.32
0.35
0.45

```
1 puffin.select(pl.col('weight').sum())
```

shape: (1, 1)

weight
f64
1.12

Expressions in group-by: a step forward

Quansight Labs

What happens if we do `pl.col('weight').sum()` inside group-by?

- pl.col('weight'): list[f64]
- pl.col('weight').sum(): f64

The data type...changed?

```
1 animals.group_by('name').agg(pl.col('weight'))
```

shape: (3, 2)

name	weight
str	list[f64]
"Puffin"	[0.32, 0.35, 0.45]
"Dolphin"	[3010.0, 3560.0]
"Manatee"	[490.0, 511.0]

```
1 animals.group_by('name').agg(pl.col('weight').sum())
```

shape: (3, 2)

name	weight
str	f64
"Dolphin"	6570.0
"Puffin"	1.12
"Manatee"	1001.0

Expressions in group-by: a step forward

Quansight Labs

What happens if we do `pl.col('weight').sum()` inside group-by?

- pl.col('weight'): list[f64]
- pl.col('weight').sum(): f64

The data type...changed?

wut?

```
animals.group_by('name').agg(pl.col('weight'))
```

shape: (3, 2)

name	weight
str	list[f64]
"Puffin"	[0.32, 0.35, 0.45]
"Dolphin"	[3010.0, 3560.0]
"Manatee"	[490.0, 511.0]

```
animals.group_by('name').agg(pl.col('weight').sum())
```

shape: (3, 2)

name	weight
str	f64
"Dolphin"	6570.0
"Puffin"	1.12
"Manatee"	1001.0

Practicality beats purity

Quansight Labs

The following would in theory be more consistent
- but probably not what helps users:

```
1 animals.group_by('name').agg(pl.col('weight'))
```

shape: (3, 2)

name	weight
str	list[f64]
"Puffin"	[0.32, 0.35, 0.45]
"Dolphin"	[3010.0, 3560.0]
"Manatee"	[490.0, 511.0]

```
1 animals.group_by('name').agg(pl.col('weight').sum())
```

shape: (3, 2)

name	weight
str	list[f64]
"Dolphin"	[6570.0]
"Puffin"	[1.12]
"Manatee"	[1001.0]

perfect
consistency between
pl.col('weight').sum()
in
select and group_by

Practicality beats purity

Quansight Labs

The following would in theory be more consistent
- but probably not what helps users:

```
1 animals.group_by('name').agg(pl.col('weight'))
```

shape: (3, 2)

name	weight
str	list[f64]
"Puffin"	[0.32, 0.35, 0.45]
"Dolphin"	[3010.0, 3560.0]
"Manatee"	[490.0, 511.0]

```
1 animals.group_by('name').agg(pl.col('weight').sum())
```

shape: (3, 2)

name	weight
str	f64
"Dolphin"	6570.0
"Puffin"	1.12
"Manatee"	1001.0

group_by aggregations aggregate to a single value

pandas syntax comparison

Quansight Labs

E.g. “find the maximum value of ‘c’ where ‘b’ was greater than its mean, per group ‘a’”

pandas

- either use apply and a Python lambda (slow!)
- Pre-compute `’b’`’s mean per group (i.e. do 2 group-bys)

Polars

```
df.group_by(‘a’).agg(
    pl.col(‘c’).filter(
        pl.col(‘b’) > pl.col(‘b’).mean()
    ).max()
)
```

Quansight Labs

Expressions

(can we use them in pandas?)

Expressions in pandas

Say you want to add a column `'c'`, taken by adding 1 to column 'a'

```python
import pandas as pd

df = pd.DataFrame(
    {
        "a": [1, 2, 3, 4],
        "b": [6, 2, 4, 1],
    }
)
df
```

	a	b
0	1	6
1	2	2
2	3	4
3	4	1

Expressions in pandas

Quansight Labs

Say you want to add a column `'c'`, taken by adding 1 to column 'a'

You could do:

- `df['c'] = df['a'] + 1`
- `df = df.assign(`
  `c=lambda x: x['a']+1`
  `)`

```python
import pandas as pd

df = pd.DataFrame(
    {
        "a": [1, 2, 3, 4],
        "b": [6, 2, 4, 1],
    }
)
df
```

	a	b
0	1	6
1	2	2
2	3	4
3	4	1

Expressions in pandas

Quansight Labs

Say you want to add a column `'c'`, taken by adding 1 to column 'a'

You could do:

- `df['c'] = df['a'] + 1`
-
```
df = df.assign(
        c=lambda x: x['a']+1
    )
```

Would we be able to use expressions instead?

```
df = df.assign(c=col('a')+1)
```

```python
import pandas as pd

df = pd.DataFrame(
    {
        "a": [1, 2, 3, 4],
        "b": [6, 2, 4, 1],
    }
)
df
```

	a	b
0	1	6
1	2	2
2	3	4
3	4	1

Expressions in pandas

Quansight Labs

```python
class Expr:
    def __init__(self, func):
        self._func = func

    def __add__(self, other):
        if isinstance(other, Expr):
            return Expr(lambda df: self(df) + other(df))
        return Expr(lambda df: self(df) + other)

    def __mul__(self, other):
        if isinstance(other, Expr):
            return Expr(lambda df: self(df) * other(df))
        return Expr(lambda df: self(df) * other)

    def __call__(self, df):
        return self._func(df)


def col(col_name):
    return Expr(lambda df: df.loc[:, col_name])
```

```python
print(
    df.assign(c=col("a") + 1,
              d=col("a") * col("b"))
)
```

	a	b	c	d
0	1	6	2	6
1	2	2	3	4
2	3	4	4	12
3	4	1	5	4

Can expressions bring us...together?

Quansight Labs

Narwhals

- Lightweight and extensible compatibility layer between dataframe libraries
- **Use expressions!**
- Support pandas, Polars, Dask, PyArrow, Modin, cuDF (and more!) without depending on any!

https://github.com/narwhals-dev/narwhals

Can expressions bring us...together?

Quansight Labs

Narwhals

- Used by Altair, scikit-lego, Marimo, Vegafusion, Shiny, Plotly (pending release?), and more!
- Contributions welcome!
- Join our mentored sprint at PyLadiesCon 2024!

Quansight Labs

Confusion

Conclusion

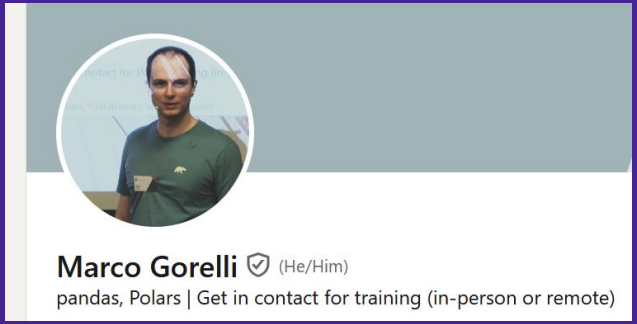
Quansight Labs

Expressions:

- Functions from Dataframes to Series
- Don’t produce values until they’re given inputs, so they allow for nice optimisations
- Polars deviates slightly from perfect consistency for the sake of user-friendliness
- There’s nothing magical about expressions, and we can re-implement them in pandas

Quansight Labs

“Too bad. Peanut butter”



Now go and use Polars expressions!

Questions?

Reach out at: connect@quansight.com

Quansight Labs

Quansight Labs

Get In Touch

connect@quansight.com