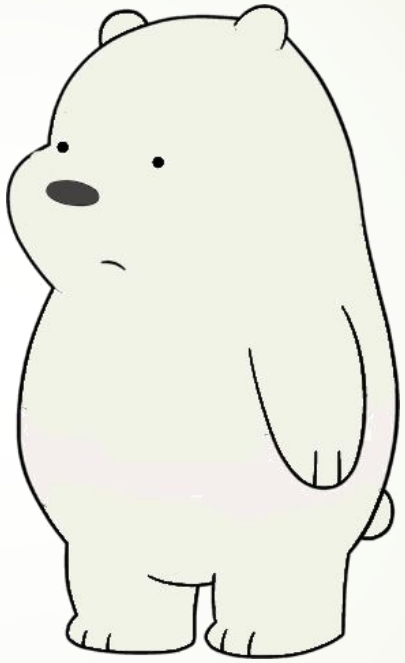
“Un tú por tú, en el análisis de datos”

Pandas Vs Polars



Mtro. Miguel Tlapa Juárez

Pythonistas GDL

- charlas
- comida
- convivencia
- sorpresas

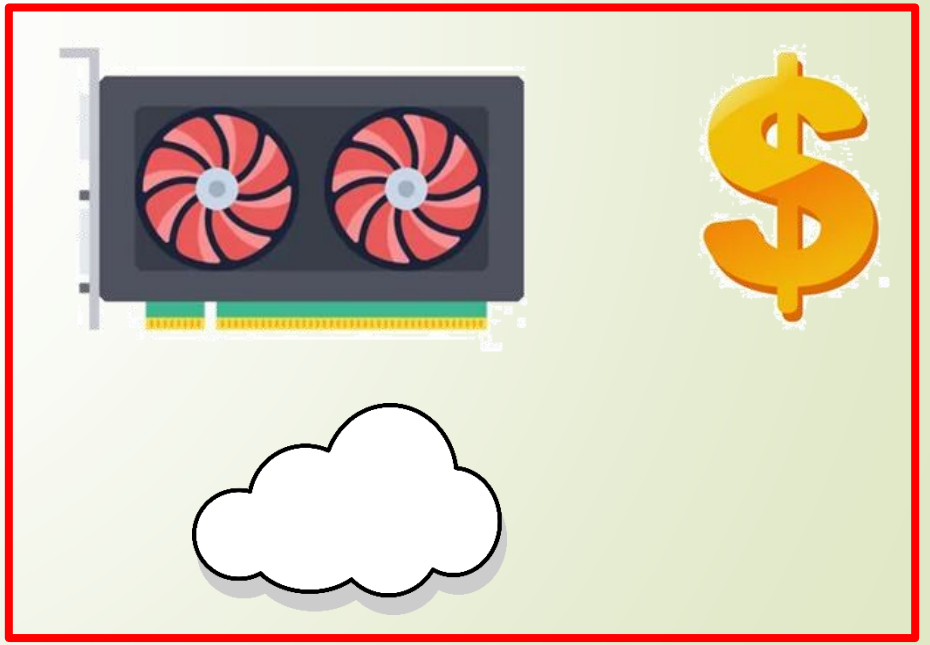
Patrocinado por

slalom

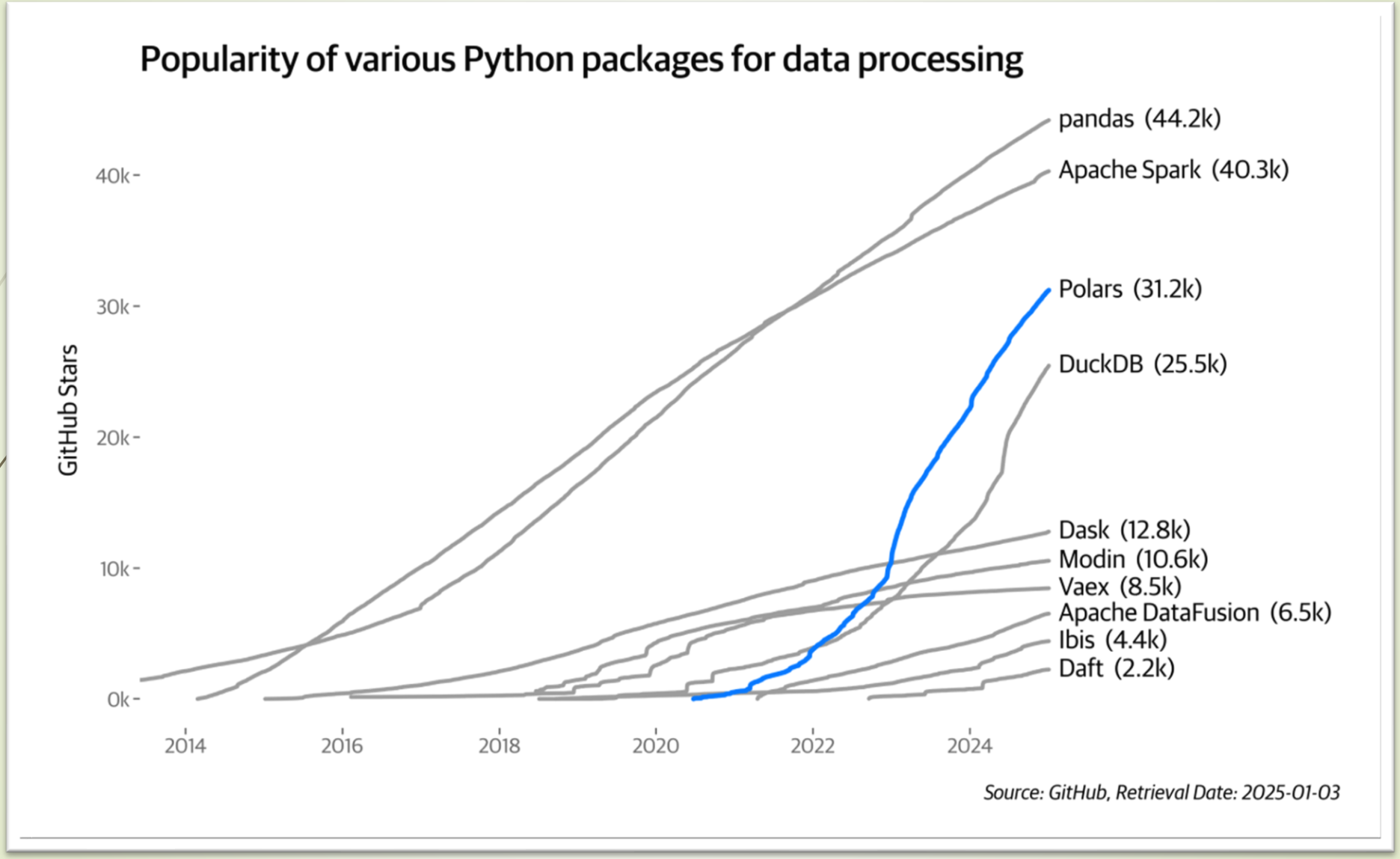
Agenda

- Definición y Contextualización del Problema
- Núcleo Funcional de Pandas
- Comparativa de Librerías
- Comparativa de Rendimiento entre Librerías
- Ejecutando Pandas y Polars
- Eager vs Lazy Frames
- Memoria RAM
- Conclusión
- Preguntas y Respuestas

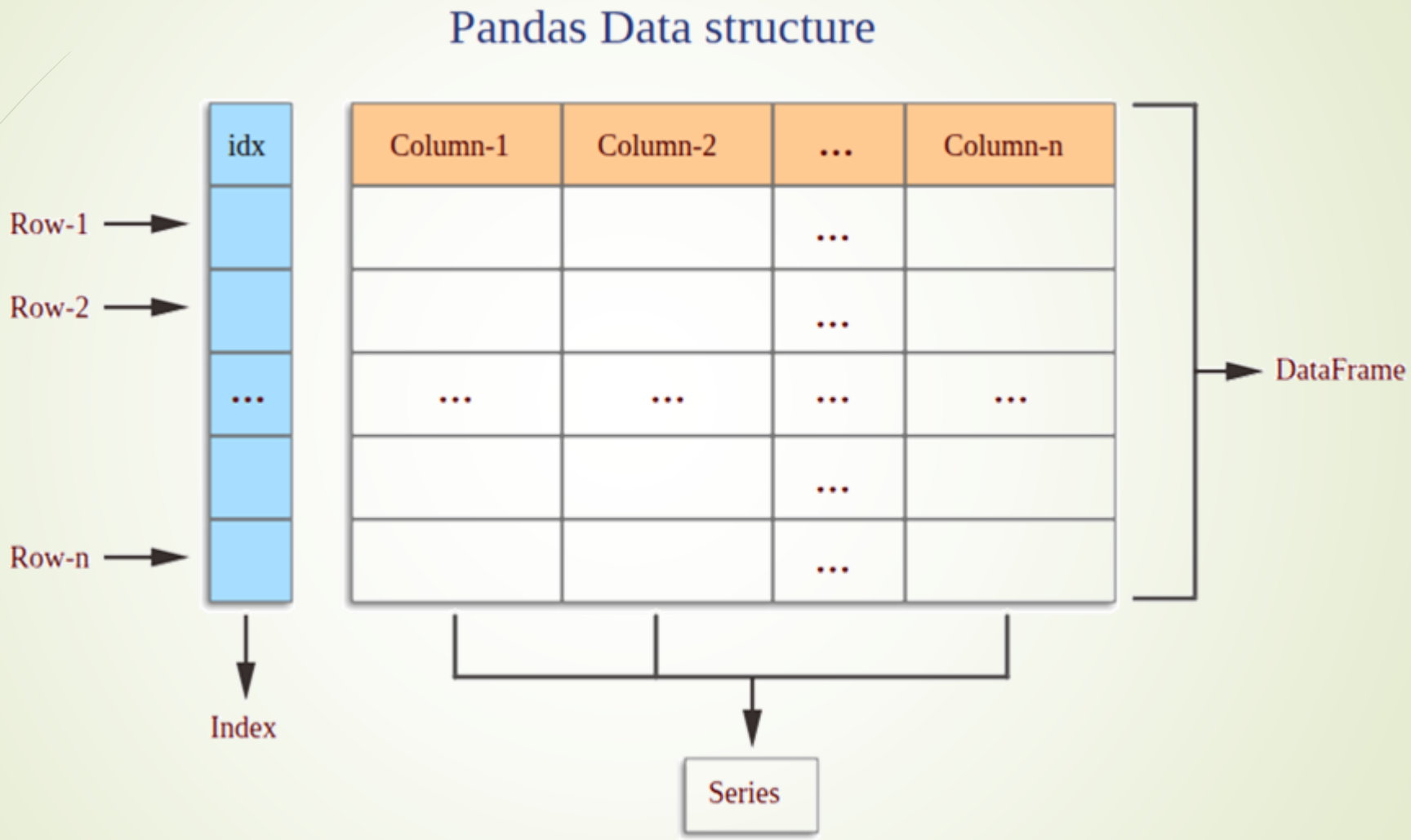
Definición y Contextualización del Problema



Definición y Contextualización del Problema



Núcleo Funcional de Pandas



Comparativa de Librerías

Software

Wes Mckinney
2008

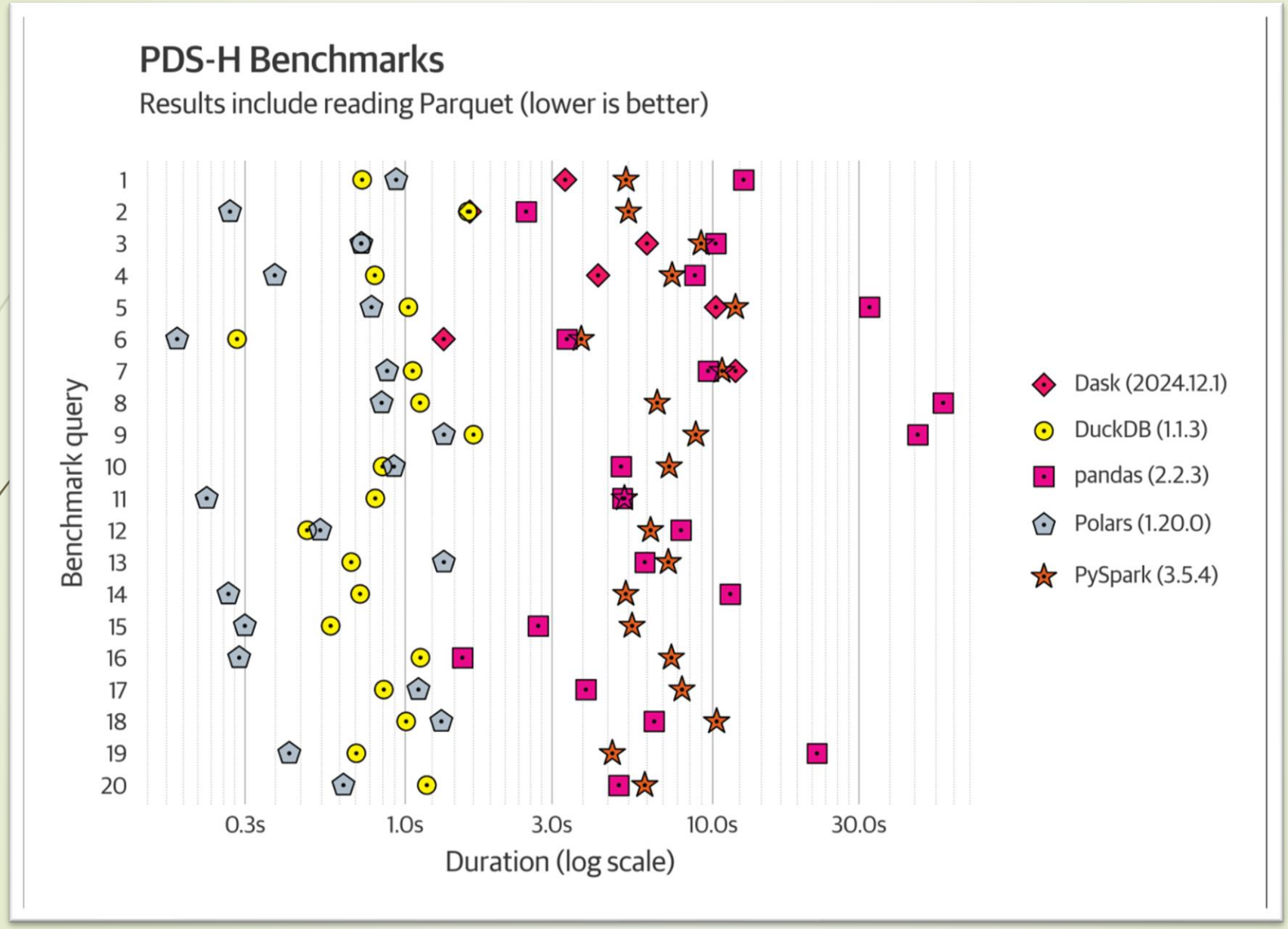
Pandas	Polars
Pandas DataFrame	Polars DataFrame
Numpy	Polar Series
Python-dateutil	Polars Datetime
Tzdata	Polars Time Zone
Analisis Temporal	Analisis Temporal
Lectura/Escritura de Archivos csv,excel,parquet	Lectura/Escritura de Archivos csv,excel,parquet
No disponible	Lazy Frame
Backend Python +C	Rust + Apache Arrow

Ritchie Vink
2020

Hardware Operations

Pandas	Polars
No usa múltiples núcleos de CPU por defecto	Ejecución Paralela Automática
No tiene soporte nativo para GPU	Diseño Eficiente en CPU
No escala bien datasets mayores a 1Gb	Escalamiento horizontal
Alto Consumo de Memoria	Bajo Consumo de Memoria

Comparativa de Rendimiento de Librerías



Ejecutando Pandas vs Polars

```
import pandas as pd
import polars as pl
✓ [1] 479ms
```

```
import sys
animals_pd = pd.read_csv("data/animals.csv", sep=",", header=0)
animals_pl = pl.read_csv("data/animals.csv", separator=",", has_header=True)
print(sys.getsizeof(animals_pd))  # Tamaño en bytes del objeto 'x'
print(sys.getsizeof(animals_pl))  # Tamaño en bytes del objeto 'x'

print(f"{type(animals_pd) = }")
print(f"{type(animals_pl) = }")
[3]
```

```
3830
48
type(animals_pd) = <class 'pandas.core.frame.DataFrame'>
type(animals_pl) = <class 'polars.dataframe.frame.DataFrame'>
```

Ejecutando Pandas vs Polars

Pandas

	animal	class	habitat	diet	lifespan	status	features	weight
0	dolphin	mammal	oceans/rivers	carnivore	40	least concern	high intelligence	150.0
1	duck	bird	wetlands	omnivore	8	least concern	waterproof feathers	3.0
2	elephant	mammal	savannah	herbivore	60	endangered	large ears and trunk	8000.0
3	ibis	bird	wetlands	omnivore	16	least concern	long, curved bill	1.0
4	impala	mammal	savannah	herbivore	12	least concern	long, curved horns	70.0
5	kudu	mammal	savannah	herbivore	15	least concern	spiral horns	250.0
6	narwhal	mammal	arctic ocean	carnivore	40	near threatened	long, spiral tusk	NaN
7	panda	mammal	forests	herbivore	20	vulnerable	black and white coloration	100.0
8	polar bear	mammal	arctic	carnivore	25	vulnerable	thick fur and blubber	720.0
9	ray	fish	oceans	carnivore	20	NaN	flat, disc-shaped body	90.0

Polars

animal	class	habitat	diet	lifespan	status	features	weight
"dolphin"	"mammal"	"oceans/rivers"	"carnivore"	40	"least concern"	"high intelligence"	150
"duck"	"bird"	"wetlands"	"omnivore"	8	"least concern"	"waterproof feathers"	3
"elephant"	"mammal"	"savannah"	"herbivore"	60	"endangered"	"large ears and trunk"	8000
"ibis"	"bird"	"wetlands"	"omnivore"	16	"least concern"	"long, curved bill"	1
"impala"	"mammal"	"savannah"	"herbivore"	12	"least concern"	"long, curved horns"	70
"kudu"	"mammal"	"savannah"	"herbivore"	15	"least concern"	"spiral horns"	250
"narwhal"	"mammal"	"arctic ocean"	"carnivore"	40	"near threatened"	"long, spiral tusk"	null
"panda"	"mammal"	"forests"	"herbivore"	20	"vulnerable"	"black and white coloration"	100
"polar bear"	"mammal"	"arctic"	"carnivore"	25	"vulnerable"	"thick fur and blubber"	720
"ray"	"fish"	"oceans"	"carnivore"	20	""	"flat, disc-shaped body"	90

Ejecutando Pandas vs Polars

Pandas

```
animals_pd["animal"]
[6]

0       dolphin
1          duck
2      elephant
3          ibis
4         impala
5          kudu
6       narwhal
7         panda
8    polar bear
9           ray
Name: animal, dtype: object
```

Polars

```
animals_pl.get_column("animal")
[7]

10 rows  10 rows × 1 cols

animal
"dolphin"
"duck"
"elephant"
"ibis"
"impala"
"kudu"
"narwhal"
"panda"
"polar bear"
"ray"
```

Ejecutando Pandas vs Polars

Pandas

```
animals_pd = animals_pd.drop(columns=["habitat", "diet", "features"])
animals_pd
[8]
```

10 rows × 5 cols

	animal	class	lifespan	status	weight
0	dolphin	mammal	40	least concern	150.0
1	duck	bird	8	least concern	3.0
2	elephant	mammal	60	endangered	8000.0
3	ibis	bird	16	least concern	1.0
4	impala	mammal	12	least concern	70.0
5	kudu	mammal	15	least concern	250.0
6	narwhal	mammal	40	near threatened	NaN
7	panda	mammal	20	vulnerable	100.0
8	polar bear	mammal	25	vulnerable	720.0
9	ray	fish	20	NaN	90.0

Polars

```
animals_pl = animals_pl.drop("habitat", "diet", "features")
animals_pl
[9]
```

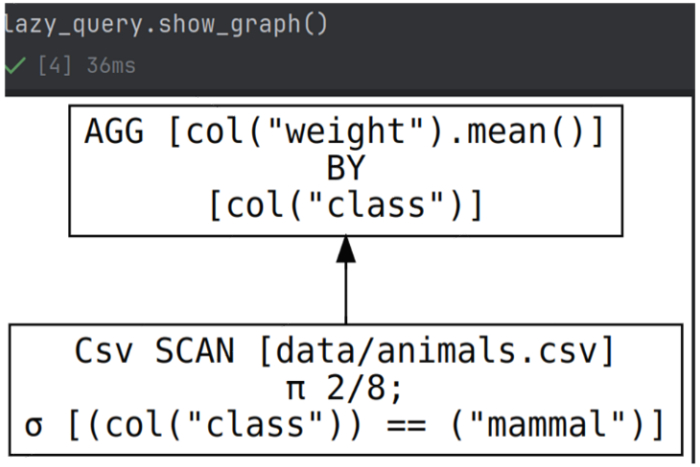
10 rows × 5 cols

animal	class	lifespan	status	weight
"dolphin"	"mammal"	40	"least concern"	150
"duck"	"bird"	8	"least concern"	3
"elephant"	"mammal"	60	"endangered"	8000
"ibis"	"bird"	16	"least concern"	1
"impala"	"mammal"	12	"least concern"	70
"kudu"	"mammal"	15	"least concern"	250
"narwhal"	"mammal"	40	"near threatened"	null
"panda"	"mammal"	20	"vulnerable"	100
"polar bear"	"mammal"	25	"vulnerable"	720
"ray"	"fish"	20	""	90

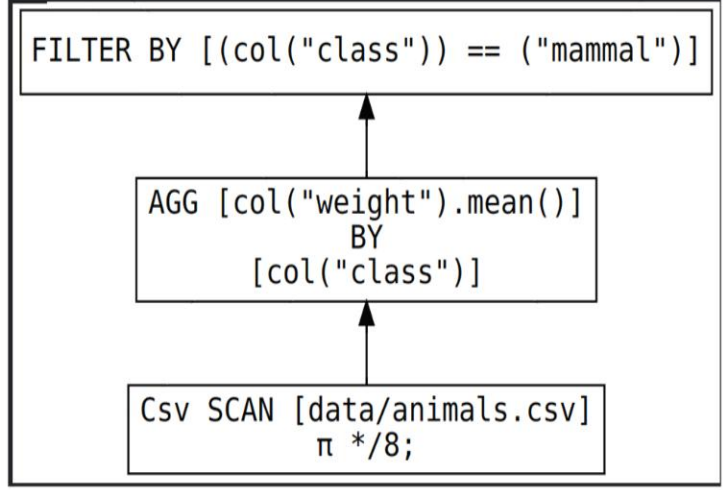
Eager vs LazyFrames

```
lazy_query = (
    pl.scan_csv("data/animals.csv")
    .group_by("class")
    .agg(pl.col("weight").mean())
    .filter(pl.col("class") == "mammal")
)
✓ [2] 16ms
```

```
lazy_query.show_graph()
✓ [4] 36ms
```




```
lazy_query.show_graph(optimized=False)
✓ [3] 67ms
```




```
lazy_query.collect()
[24]
```

1 row 1 rows × 2 cols

class	weight
"mammal"	1548.333333

Eager vs LazyFrames

```
%%time
trips = pl.read_parquet("data/taxi/yellow_tripdata_*.parquet")
sum_per_vendor = trips.group_by("VendorID").sum()
income_per_distance_per_vendor = sum_per_vendor.select(
    "VendorID",
    income_per_distance=pl.col("total_amount") / pl.col("trip_distance"),
)
top_three = income_per_distance_per_vendor.sort(
    by="income_per_distance", descending=True
).head(3)

top_three

✓ [3] 1s 274ms
CPU times: user 10.2 s, sys: 2.89 s, total: 13 s
Wall time: 1.27 s
```

3 rows | 3 rows × 2 cols

VendorID	income_per_distance
1	6.434789
6	5.296493
5	4.731557

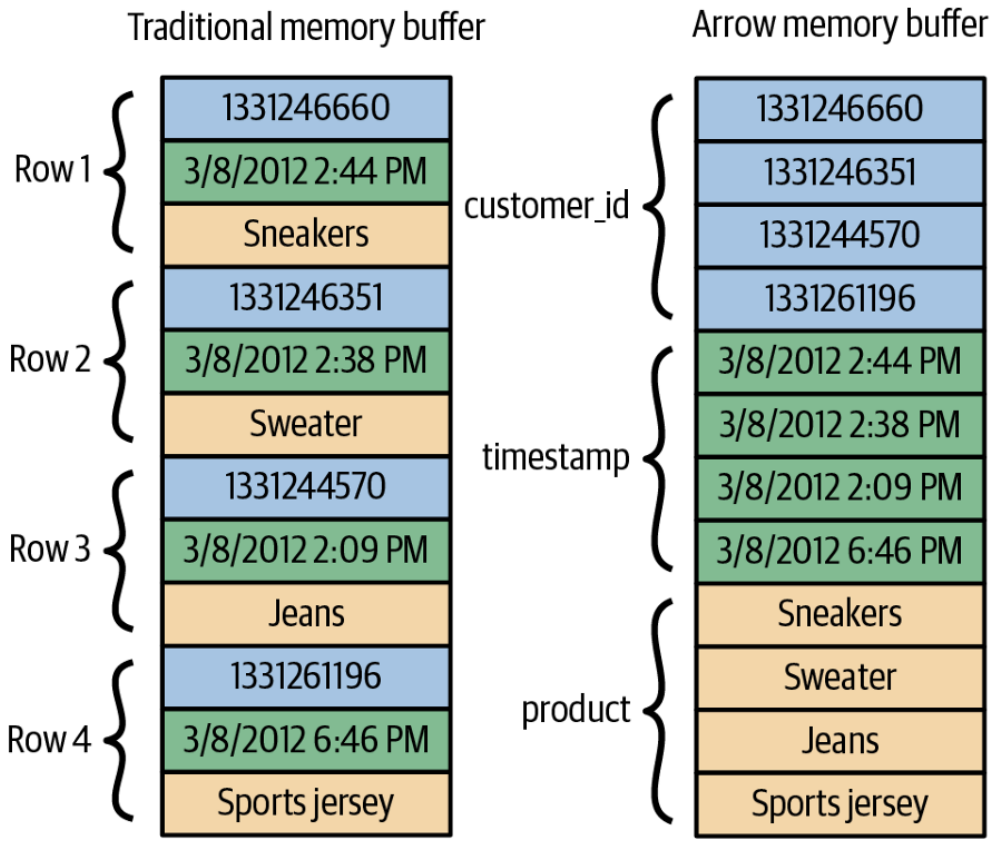
Eager vs LazyFrames

```
%%time
trips = pl.scan_parquet("data/taxi/yellow_tripdata_*.parquet")
sum_per_vendor = trips.group_by("VendorID").sum()

income_per_distance_per_vendor = sum_per_vendor.select(
    "VendorID",
    income_per_distance=pl.col("total_amount") / pl.col("trip_distance"),
)

top_three = income_per_distance_per_vendor.sort(
    by="income_per_distance", descending=True
).head(3)
print(top_three.describe())
top_three.collect()
```

```
┌────────────┬──────────┬──────────┐
│ str        ┆ f64      ┆ f64      │
╞════════════╪══════════╪══════════╡
│ count      ┆ 3.0      ┆ 3.0      │
│ null_count ┆ 0.0      ┆ 0.0      │
│ mean       ┆ 4.0      ┆ 5.487613 │
│ std        ┆ 2.645751 ┆ 0.867551 │
│ min        ┆ 1.0      ┆ 4.731557 │
│ 25%        ┆ 5.0      ┆ 5.296493 │
│ 50%        ┆ 5.0      ┆ 5.296493 │
│ 75%        ┆ 6.0      ┆ 6.434789 │
│ max        ┆ 6.0      ┆ 6.434789 │
└────────────┴──────────┴──────────┘
CPU times: user 3.94 s, sys: 46.6 ms, total: 3.99 s
Wall time: 432 ms
```

Memoria RAM

	customer_id	timestamp	product
Row 1	1331246660	3/8/2012 2:44 PM	Sneakers
Row 2	1331246351	3/8/2012 2:38 PM	Sweater
Row 3	1331244570	3/8/2012 2:09 PM	Jeans
Row 4	1331261196	3/8/2012 6:46 PM	Sports jersey



Conclusión

- Pandas es ideal para tareas educativas, prototipos rápidos, análisis exploratorio con data sets no mayores a 1Gb
- Polars es mejor en rendimiento para grandes volúmenes de datos con mejoras en operaciones de filtrado y manipulación de datos
- Sin embargo se puede adoptar una estrategia hibrida.

Preguntas y Respuestas

Redes Sociales

migueltlapa

@MIGUELTLAPA