A quack of all trades:

DuckDB, a versatile analytical database to keep in your toolkit

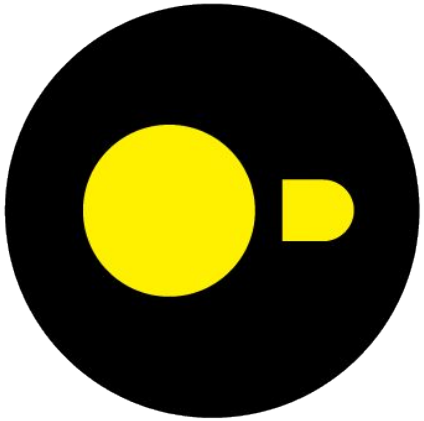
Ned Letcher

Simon Aubury

DATA ENG Australia

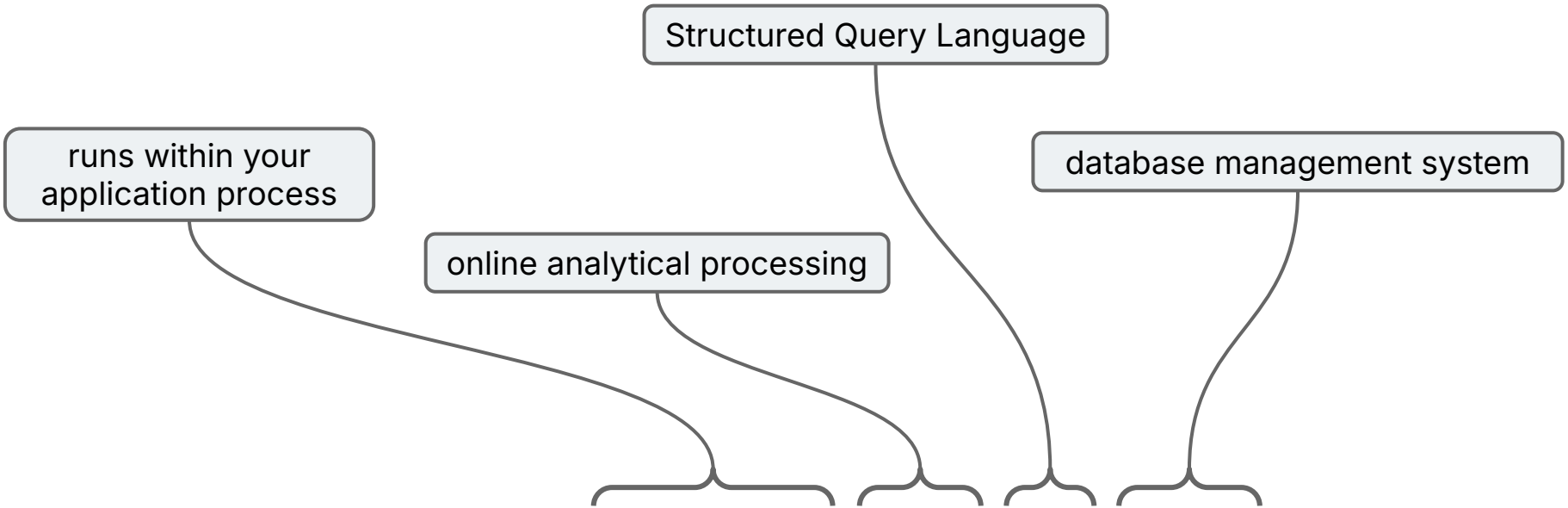
https://en.wikipedia.org/wiki/Get_a_Mac

OK, so what is DuckDB?



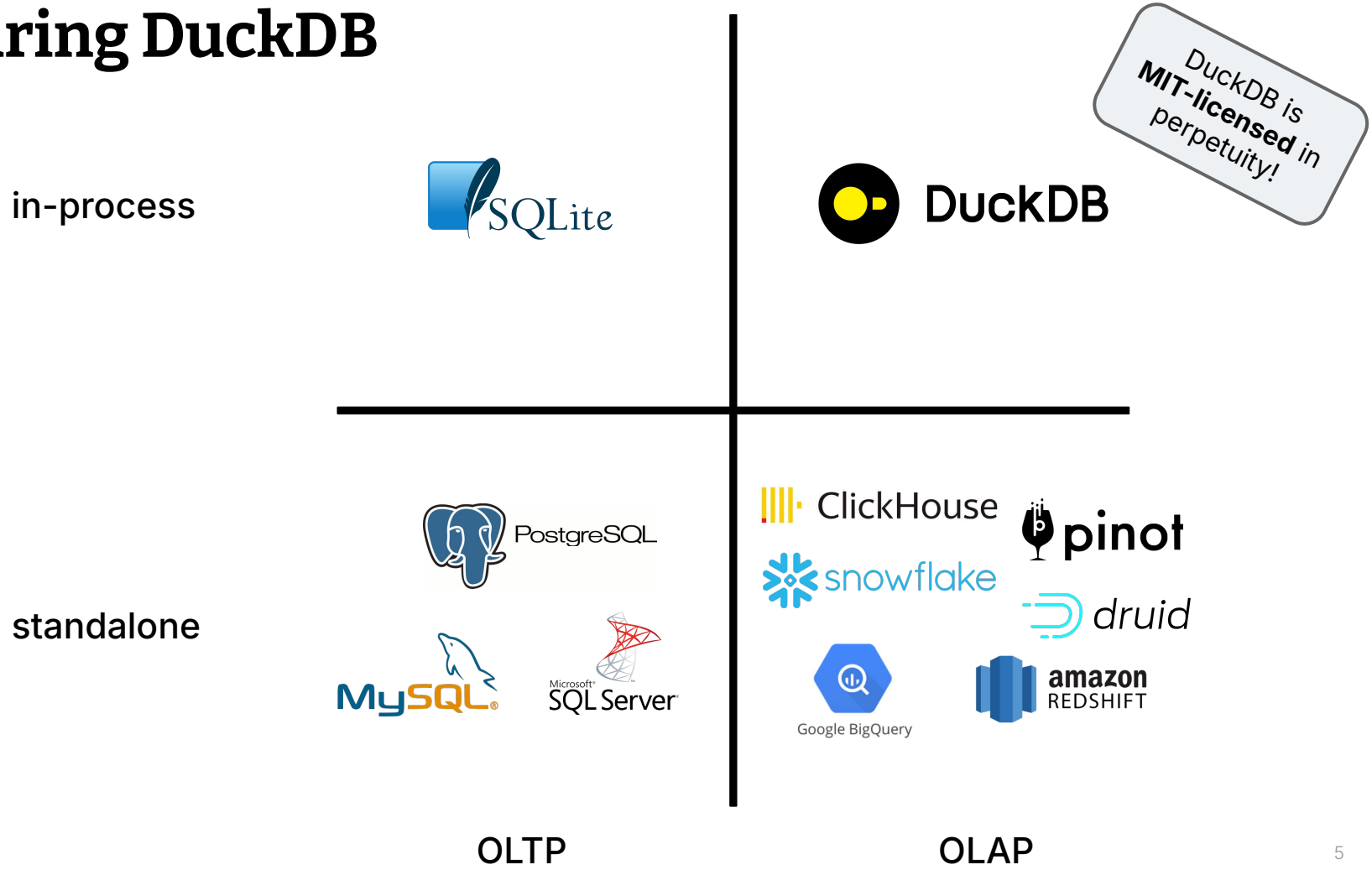
DuckDB is a fast in-process analytical database

OK, so what is DuckDB?

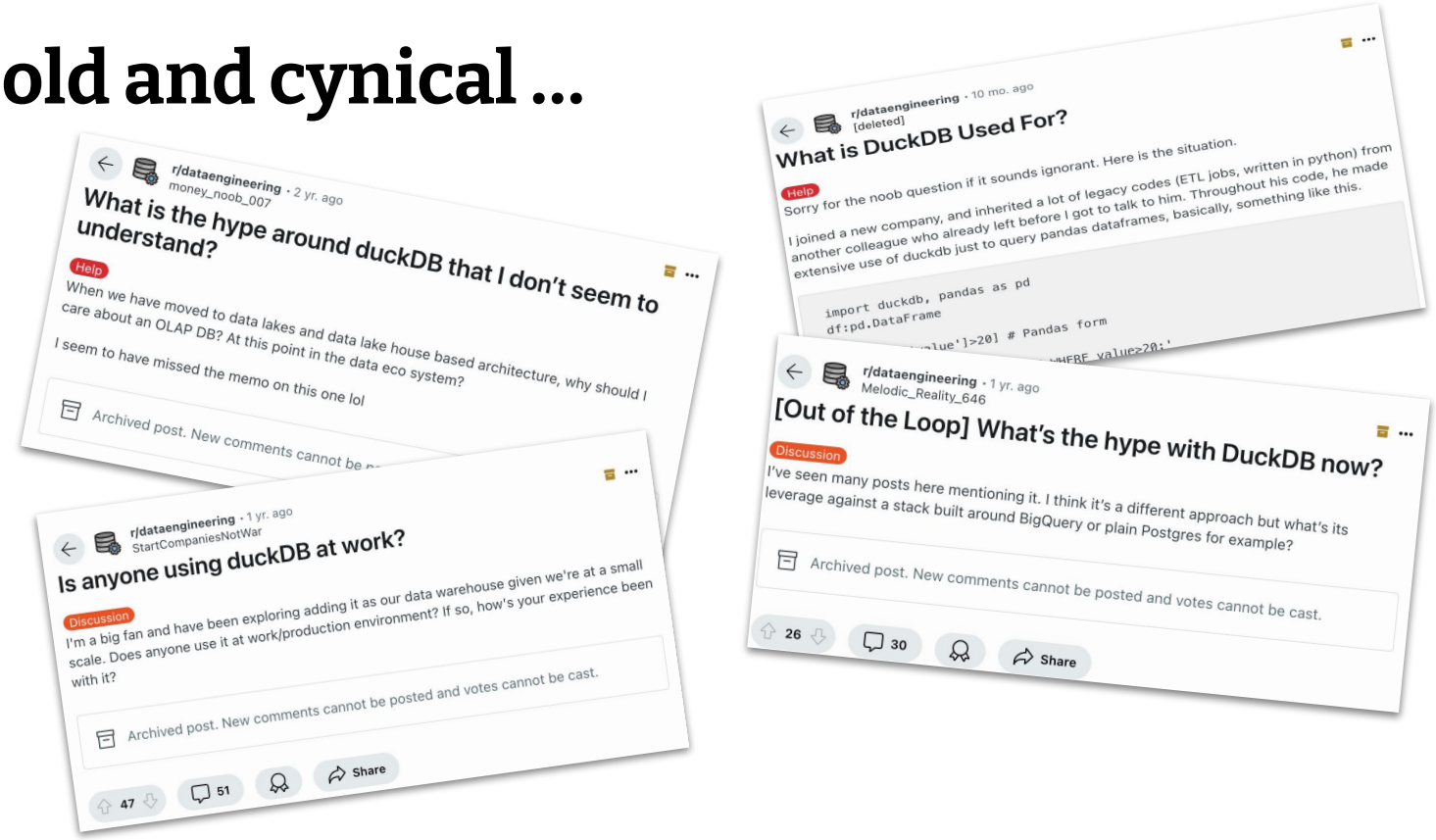


DuckDB is an in-process OLAP SQL DBMS.

Comparing DuckDB

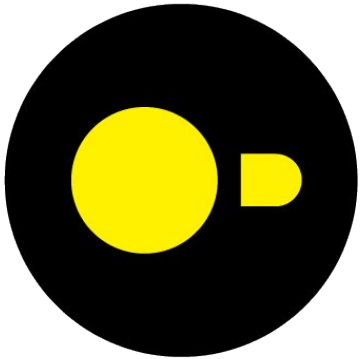


I’m old and cynical …

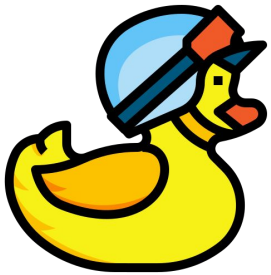


https://www.reddit.com/r/dataengineering

DuckDB is Versatile



- Data analysis
- Data pipelines
- Data lake querying
- Data warehousing
- Data cubes
- Data apps



DuckDB for data engineers

Demo CSV files

pizza_n.csv

```
food_name,color,calories,is_healthy
Margherita,mixed,60,FALSE
Chicago,yellow,70,FALSE
Pep
```

```
food_name,color,calories,is_healthy
Hawaiian,yellow,60,FALSE
Neapol
```

```
food_name,color,calories,is_healthy
BBQ,red,62,FALSE
Veggie,mixed,32,TRUE
```

fast_food.csv

```
food_name,color,calories
burger,mixed,60
fries,yellow,35
sandwhich,white,22
```

salad.csv

```
food_name,calories,is_healthy,color
sushi,60,TRUE,mixed
pho,70,TRUE,yellow
```

Names & types

```
-- Mixed CSV schemas - this will adapt as the schema changes
SELECT *
FROM read_csv('./data_food/*.csv', union_by_name=true);
```

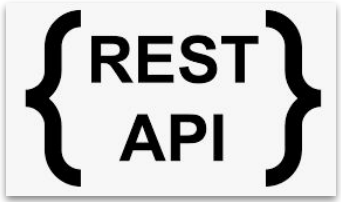
food_name varchar	calories int64	is_healthy boolean	color varchar
sushi	60	true	mixed
pho	70	true	yellow
burger	60		mixed
fries	35		yellow
sandwhich	22		white
Margherita	60	false	mixed
Chicago	70	false	yellow
Pepperoni	55	false	red
Hawaiian	60	false	yellow
Neapolitan	45	false	red
BBQ	62	false	red
Veggie	32	true	mixed
salad	50	true	green
yogurt	20	true	white

Anywhere


```sql
-- Reading remote files with the httpfs extension
CREATE OR REPLACE SECRET mysecret (
    TYPE S3,
    REGION 'us-east-1',
    ENDPOINT 's3.amazonaws.com'
);

SELECT *
FROM read_parquet('s3://duckdb-s3-bucket-public/countries.parquet')
WHERE name SIMILAR TO '.*Republic.*';
```

```sql
SELECT json_extract(hourly, '$.temperature_2m[1]')
from read_json('https://api.open-meteo.com/v1/forecast?
latitude=-33.8678&longitude=151.2073&hourly=temperature_2m&forecast_days=1');
```




```sql
INSTALL sqlite_scanner;
LOAD sqlite_scanner;

ATTACH './data_iMessage/chat.db' as imessage_chat_sqlite (TYPE sqlite);

SELECT text
FROM imessage_chat_sqlite.message;
```

A mini lakehouse w., Parquet

Projection pushdown into the Parquet file itself for columns

Hive partitioning filters on the partition keys

Filter pushdown for Parquet zone-maps

```
SELECT count(*), max(ride_amt)
FROM read_parquet('s3://mybucket/taxi/*/*/*.parquet',
  hive_partitioning = true)
WHERE year = 2024 AND month = 9
AND passengers > 4;
```

https://duckdb.org/docs/data/parquet/overview.html

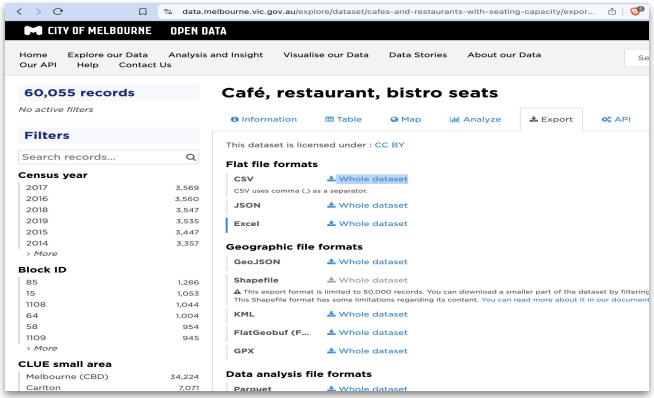
DuckDB is extensible

https://duckdb.org/docs/extensions/overview.html

Complex remote

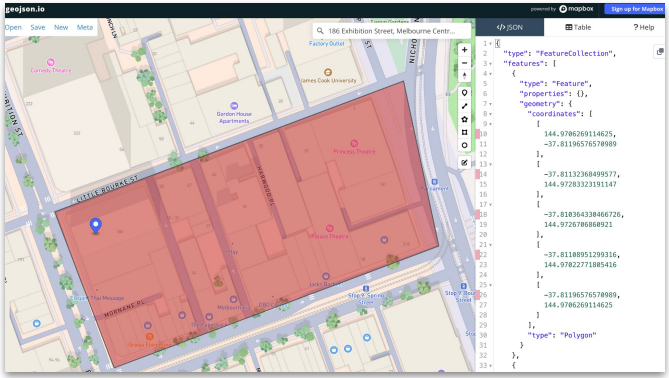
Remote file

https://data.melbourne.vic.gov.au



Geo-json

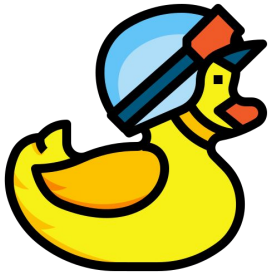
https://geojson.io




```
-- Files don't need to be local
SELECT *
FROM read_csv(
'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/d
capacity/exports/csv?lang=en&timezone=Australia%2FSydney&use_
) LIMIT 10;
```

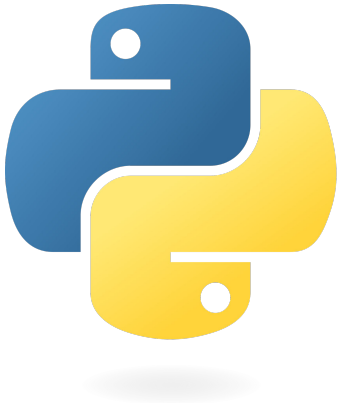
```
SELECT *
FROM cafes
WHERE st_within(
    st_point(longitude, latitude),
    (
        SELECT geom
        FROM st_read('data_geo/DEB_melborne_boundary_region.geojson')
    )
)
AND "Industry (ANZSIC4) description" = 'Cafes and Restaurants'
AND "Seating type" = 'Seats - Indoor'
AND "Number of seats" < 30
```

Trading name	Business address	Seating type	Number of seats
Shimbashi Japanese Soba & Sake Bar	17 Liverpool Street MELBOURNE 3000	Seats - Indoor	24
Shimbashi Japanese Soba & Sake Bar	17 Liverpool Street MELBOURNE 3000	Seats - Indoor	24
Butchers Diner	Ground 10 Bourke Street MELBOURNE VIC 3000	Seats - Indoor	20
Rice Paper Scissors Asian Kitchen	19 Liverpool Street MELBOURNE 3000	Seats - Indoor	25



DuckDB for data scientists

You can use DuckDB with the tools you already use



Python client

R client

Streamlined user experience

```
$ ls -1 ./data

174502796.json
17450310.json
17451211.json
17451321.json
...
```

In-memory data wrangling

```
import duckdb
duckdb.sql(
    """
    SELECT id, name, filename
    FROM read_json('data/*.json', filename=true)
    """
).to_parquet("users.parquet")
```

```
$ pip install duckdb
```

Loading into a persistent database

```
conn = duckdb.connect("my.duckdb")

conn.sql(
    """
    CREATE OR REPLACE TABLE users AS
    SELECT id, name, filename
    FROM read_json('data/*.json', filename=true)
    """
)

conn.sql("SELECT count(*) FROM users")
```

Friendly SQL

```
SELECT * EXCLUDE (city)
FROM addresses;
```

```
SELECT * REPLACE (lower(city) AS city)
FROM addresses;
```

```
SELECT COLUMNS('number\d+')
FROM addresses;
```

Column selection expressions

```
FROM tbl
SELECT i, s;
```

```
FROM tbl;
```

FROM-first syntax

```
SELECT [
    lower(x) FOR x IN ['Hello', '', 'World']
    IF len(x) > 0
] AS strings;
```

List comprehensions

```
SELECT
    42 AS x,
    ['a', 'b', 'c',] AS y,
    'hello world' AS z,
;
```

Trailing commas

Performance

DuckDB allows you to scale the size of your workloads on your laptop further, before you need a distributed compute solution.

- Vectorized query execution model
- Multi-core processing
- Out-of-core processing
- Compression
- Lazy evaluation
- Query optimiser

Lazy evaluation

- **relations**: query representations
- Evaluated as needed
- Compose with **replacement scans**
- Enables query optimization

```
relation = conn.sql("FROM users USING SAMPLE 100000")
```

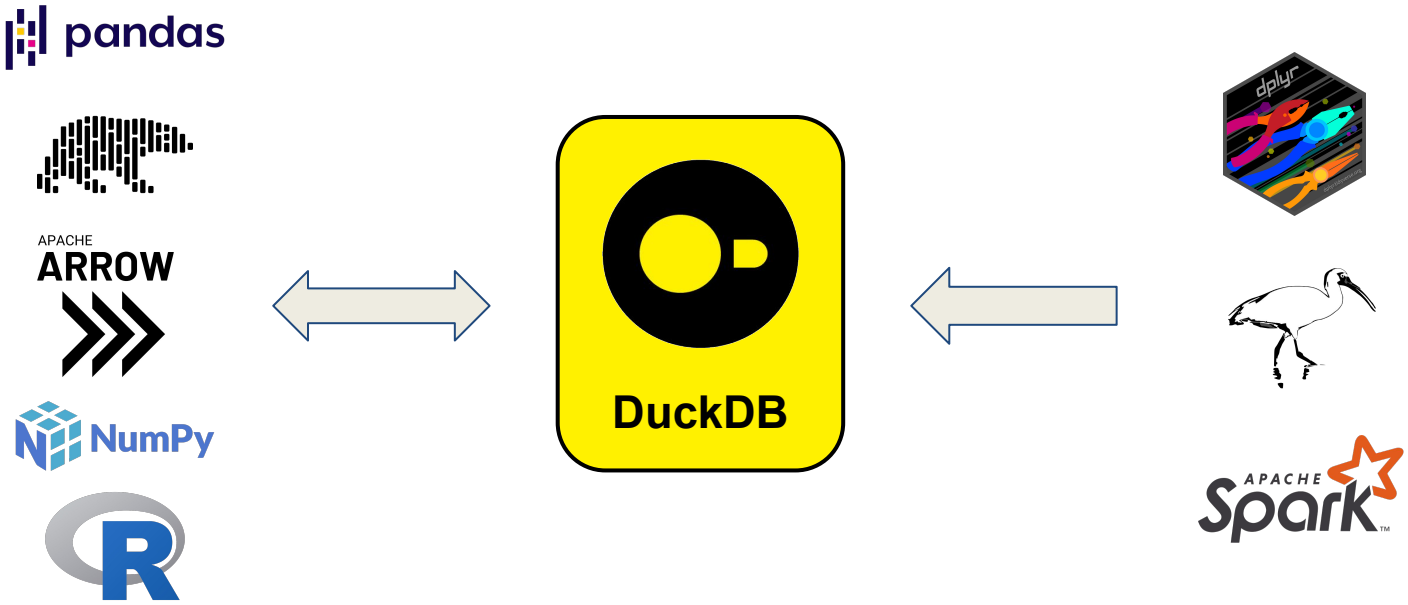
```
relation.show()
```

```
┌─────────────┬──────────────────────┬──────────────────────────────────────────────┐
│     id      │         name         │                   filename                   │
│   varchar   │       varchar        │                    varchar                   │
├─────────────┼──────────────────────┼──────────────────────────────────────────────┤
│ 735065622   │ JJ Gato              │ data/12144232_following.json                 │
│ 10793512    │ Laura Bright, Ph.D.  │ data/12687952_following.json                 │
│ 133026272   │ Stuff to Blow Your Mind │ data/123085589_following.json             │
│     .       │          .           │                    .                         │
│     .       │          .           │                    .                         │
│     .       │          .           │                    .                         │
│ 2236341775  │ FGI France           │ data/1206374840_following.json               │
│ 330556991   │ Spencer Muhlstock    │ data/12369372_following.json                 │
│ 67380876    │ Shailesh Shukla      │ data/1260881050005315586_following.json      │
├─────────────┴──────────────────────┴──────────────────────────────────────────────┤
│ ? rows (>9999 rows, 6 shown)                                          3 columns │
└─────────────────────────────────────────────────────────────────────────────────┘
```

```
conn.sql("SELECT mean(length(name)) FROM relation")
```

```
┌──────────────────────────┐
│ mean(length("name"))     │
│          double          │
├──────────────────────────┤
│                 14.01666 │
└──────────────────────────┘
```

Integration with data tooling

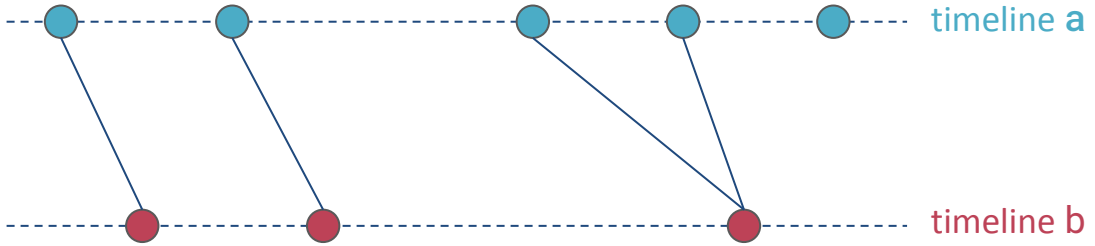


In-memory data formats

Alternative DuckDB query interfaces

Versatility over data types

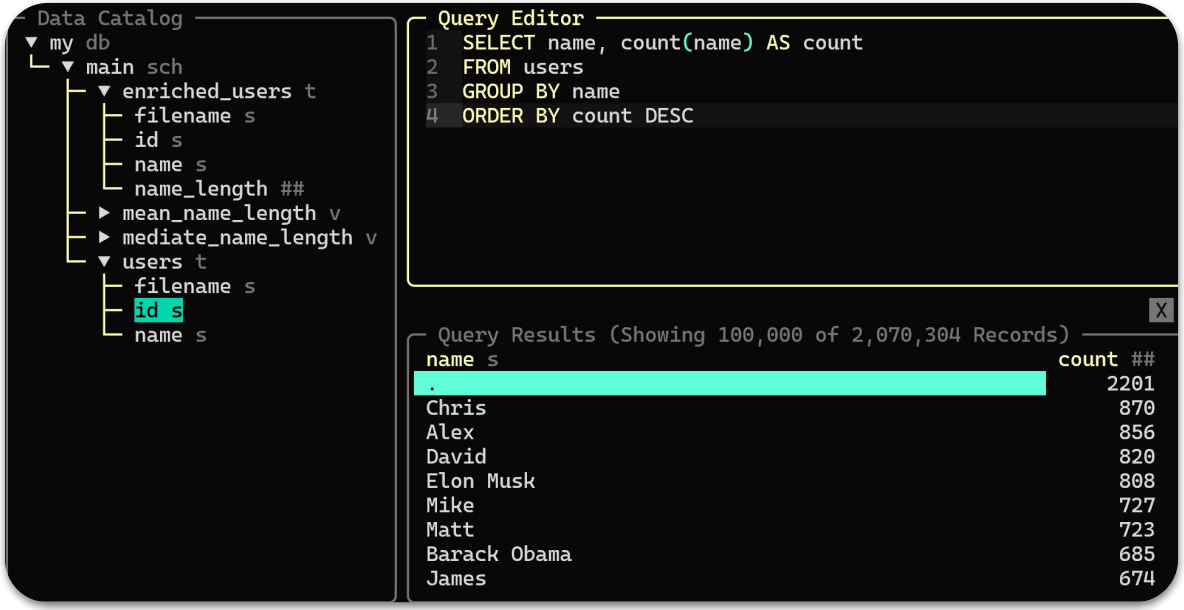
- High-cardinality data
- Multidimensional data
- **Time series data**
- Semi-structured data
- Spatial data
- Vector data
- Text data




```
SELECT a.id, a.when, price
FROM a
ASOF JOIN b
       ON a.id = b.id
      AND a.when >= b.when;
```

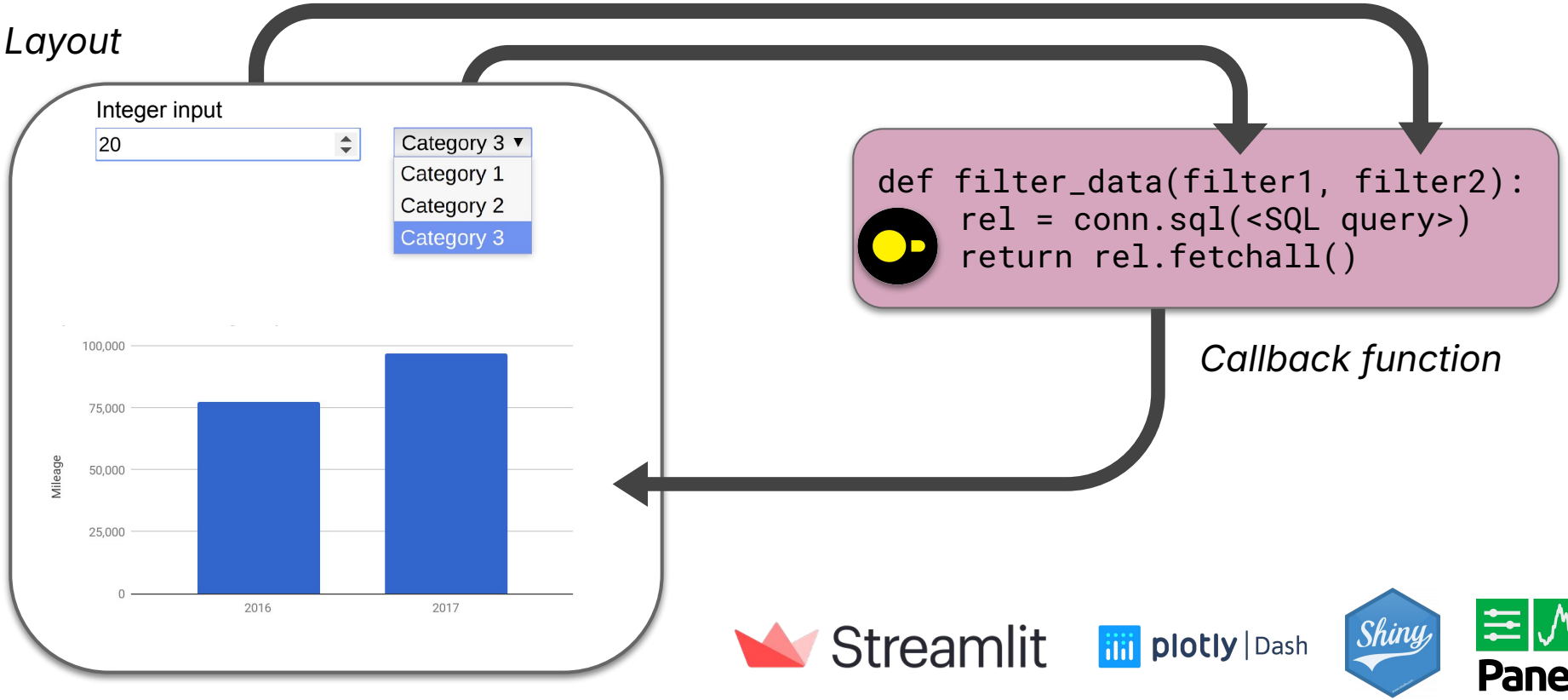
Data management

- ACID transactions
- Concurrency control
- Integrity constraints
- Indexes & query optimisation
- Views & CTEs
- Data catalogue



Harlequin SQL IDE

Powering data apps

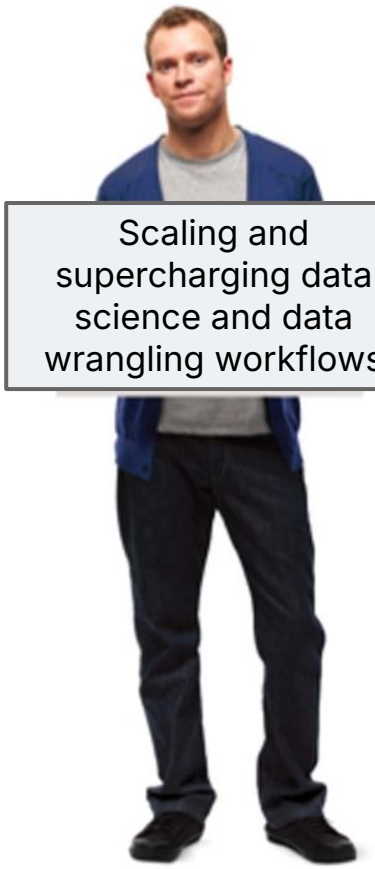


In conclusion …

Why add DuckDB to your data toolkit?

A building block for data infrastructure & data products

- 🦆 Data analysis
- 🦆 Data pipelines
- 🦆 Data lake querying
- 🦆 Data warehousing
- 🦆 Data cubes
- 🦆 Data apps



Scaling and supercharging data science and data wrangling workflows

Thank you!

https://packt.link/byKYt