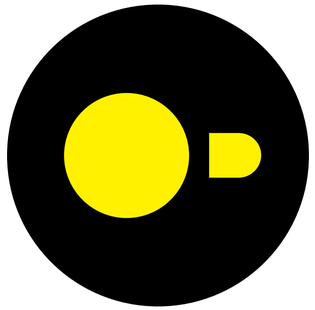


Data Wrangling Like a Boss With DuckDB

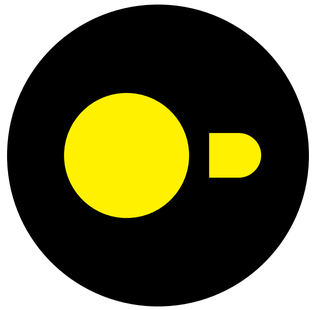
Hannes Mühleisen

DuckDB

Co-Founder & CEO

Professor

Item	Amount
Hop	42
Barley	84



If your data fits in memory there is no advantage to putting it in a database: it will only be **slower** and more **frustrating.**

CSV files for download

The CSV files on this page contain the latest data from Infoshare and our information releases. 2013 Census meshblock data is also available in CSV format.

You can download CSV files about entire Infoshare subjects. This saves you downloading multiple files from Infoshare. We publish CSV files with most information releases, and add them to this page once the release is published.

The files are grouped by topic:

- Business
- Census
- Economy
- Environment
- Government finance

Census

2018 Census totals by topic – national highlights – CSV

ZIP compressed file, 68 KB

Statistical area 1 dataset for 2018 Census – web page includes dataset in Excel and CSV format, footnotes, and other supporting information

Age and sex by ethnic group (grouped total responses), for census night population counts, 2006, 2013, and 2018 Censuses (RC, TA, SA2, DHB), CSV zipped file, 98 MB

Age and sex by ethnic group (grouped total responses), for census usually resident population counts, 2006, 2013, and 2018 Censuses (RC, TA, SA2, DHB), CSV zipped file, 103 MB

Occupied dwellings, unoccupied dwellings, and dwellings under construction, for private and non-private dwellings, 2006, 2013, and 2018 Censuses (RA, TA, SA2, DHB), CSV zipped file, 537 KB

2013 Census meshblock dataset – contains counts at meshblock and other geographic levels for selected variables from the 2013, 2006, and 2001 Censuses.

“More Frustrating”

```
CREATE TABLE …;
COPY …;
```

```
$ head Data8277.csv
Year,Age,Ethnic,Sex,Area,count
2018,000,1,1,01,795
2018,000,1,1,02,5067
2018,000,1,1,03,2229
2018,000,1,1,04,1356
```

```
CREATE TABLE Data8277 (
  Year    INTEGER,
  Age     INTEGER,
  Ethnic INTEGER,
  Sex     INTEGER,
  Area    INTEGER,
  count   INTEGER);
```

COPY Data8277 FROM '.../Data8277.csv' (FORMAT CSV, HEADER true);

ERROR: invalid input syntax for type integer: "..C"
CONTEXT: COPY data8277, line 18, column count: "..C"

```
# DROP TABLE Data8277; -- grr

# CREATE TABLE Data8277 (
    Year   INTEGER,
    Age    INTEGER,
    Ethnic INTEGER,
    Sex    INTEGER,
    Area   INTEGER,
    count  VARCHAR); -- fine

# COPY Data8277 FROM '.../Data8277.csv' (FORMAT CSV, HEADER true);
ERROR:  invalid input syntax for type integer: "CMB07601"
CONTEXT:  COPY data8277, line 431741, column area: "CMB07601"
```

```
# DROP TABLE Data8277; -- grrrrr

# CREATE TABLE Data8277 (
    Year   VARCHAR,
    Age    VARCHAR,
    Ethnic VARCHAR,
    Sex    VARCHAR,
    Area   VARCHAR,
    count  VARCHAR); -- wtf

# COPY Data8277 FROM '.../Data8277.csv' (FORMAT CSV, HEADER true);
COPY 34959672
Time: 16.36 s
```

“More Frustrating”

“Slower”

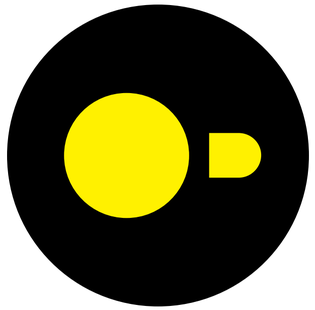
```
> readr::read_csv("Data8277.csv")

Rows: 34959672 Columns: 6
── Column specification ──
Delimiter: ","
chr (3): Age, Area, count
dbl (3): Year, Ethnic, Sex

# A tibble: 34,959,672 × 6
    Year Age    Ethnic   Sex Area  count
   <dbl> <chr>  <dbl> <dbl> <chr> <chr>
 1  2018 000        1     1 01    795
 2  2018 000        1     1 02    5067
 3  2018 000        1     1 03    2229
 4  2018 000        1     1 04    1356
 5  2018 000        1     1 05    180
# ℹ 34,959,667 more rows
```

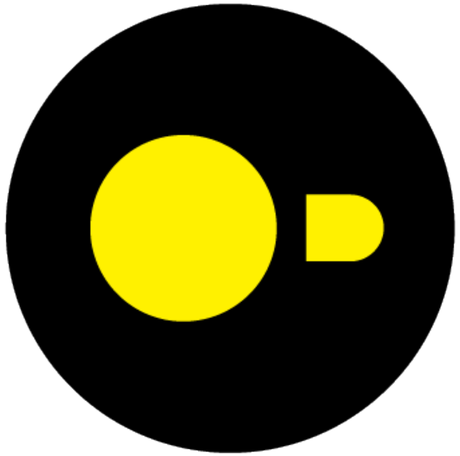
10 s

System	Command	Time
R read.csv	read.csv("Data8277.csv")	22 s
PostgreSQL	COPY … FROM …	16 s
readr	readr::read_csv("Data8277.csv")	10 s
Pandas	pandas.read_csv("Data8277.csv")	5.5 s
data.table	fread("Data8277.csv")	2.9 s
Polars	polars.read_csv("Data8277.csv")	Exception

“More Frustrating” 

“Slower” 

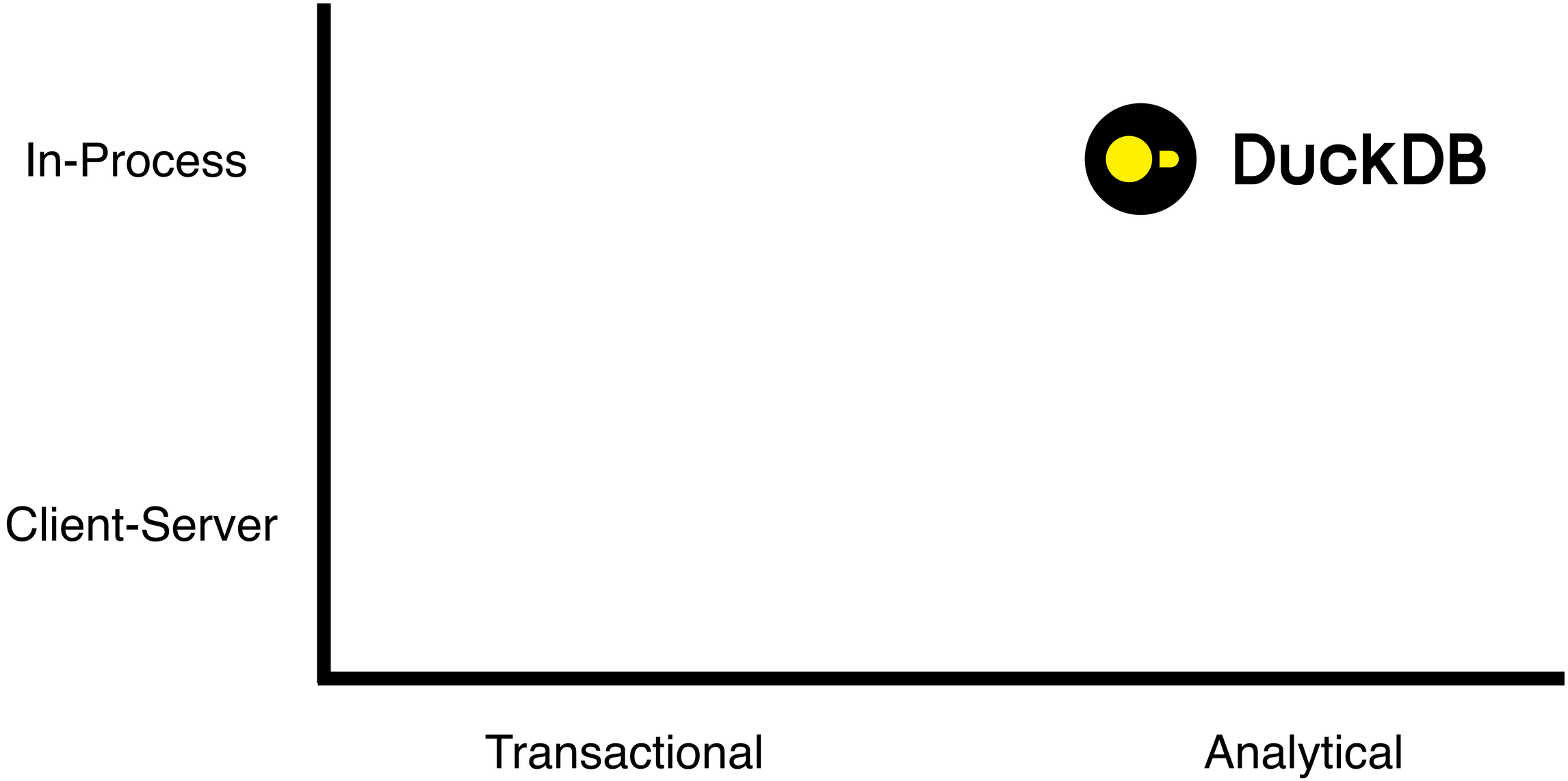


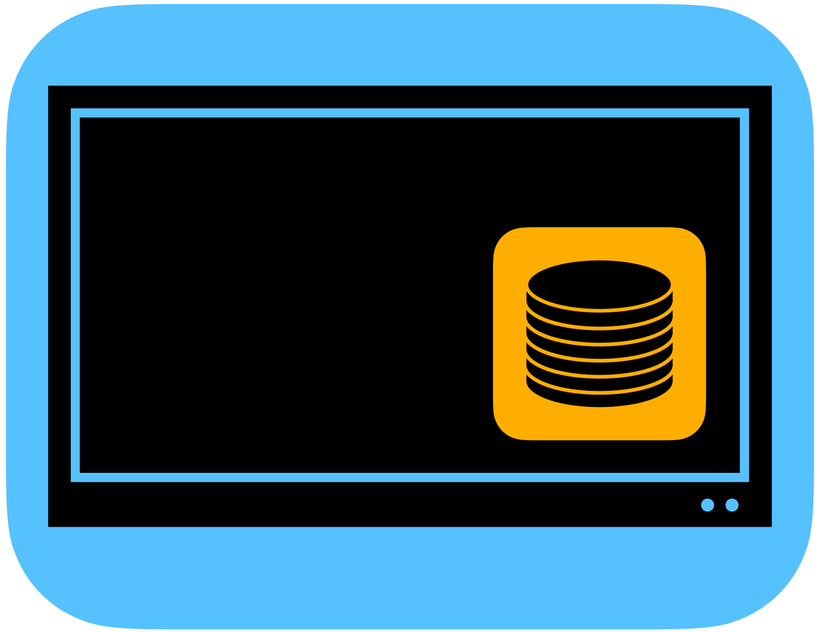


DuckDB

User
Experience

DBMS
Engine


In-Process
Client-Server
DuckDB
Transactional
Analytical

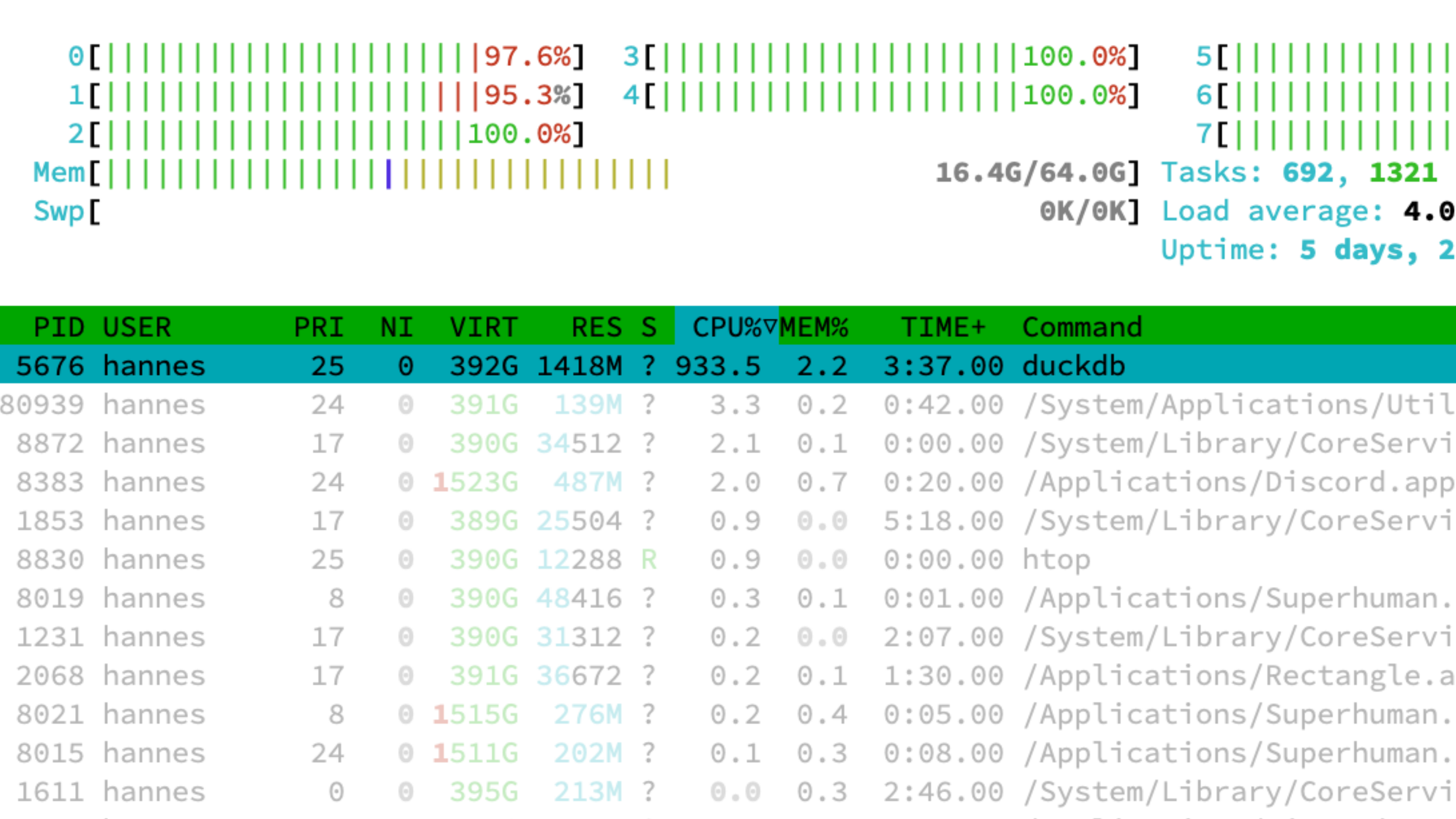


```
install.packages("duckdb")
```

```
pip install duckdb
```

Batteries Included

Fast


0[||||||||||||||||||||||97.6%]
1[||||||||||||||||||||||95.3%]
2[||||||||||||||||||||||100.0%]
3[||||||||||||||||||||||100.0%]
4[||||||||||||||||||||||100.0%]
5[||||||||||||
6[||||||||||||
7[||||||||||||
Mem[||||||||||||||||||||||||
16.4G/64.0G]
Swp[
0K/0K]
Tasks: 692, 1321
Load average: 4.0
Uptime: 5 days, 2
PID USER PRI NI VIRT RES S CPU%▽MEM% TIME+ Command
5676 hannes 25 0 392G 1418M ? 933.5 2.2 3:37.00 duckdb
80939 hannes 24 0 391G 139M ? 3.3 0.2 0:42.00 /System/Applications/Util
8872 hannes 17 0 390G 34512 ? 2.1 0.1 0:00.00 /System/Library/CoreServi
8383 hannes 24 0 1523G 487M ? 2.0 0.7 0:20.00 /Applications/Discord.app
1853 hannes 17 0 389G 25504 ? 0.9 0.0 5:18.00 /System/Library/CoreServi
8830 hannes 25 0 390G 12288 R 0.9 0.0 0:00.00 htop
8019 hannes 8 0 390G 48416 ? 0.3 0.1 0:01.00 /Applications/Superhuman.
1231 hannes 17 0 390G 31312 ? 0.2 0.0 2:07.00 /System/Library/CoreServi
2068 hannes 17 0 391G 36672 ? 0.2 0.1 1:30.00 /Applications/Rectangle.a
8021 hannes 8 0 1515G 276M ? 0.2 0.4 0:05.00 /Applications/Superhuman.
8015 hannes 24 0 1511G 202M ? 0.1 0.3 0:08.00 /Applications/Superhuman.
1611 hannes 0 0 395G 213M ? 0.0 0.3 2:46.00 /System/Library/CoreServi

Alexander Doria
@Dorialexander

AI aside, @duckdb is probably the most magical piece of technology of recent years.

8:37 AM · Aug 14, 2024 · **5,312** Views

Task

groupby | join

0.5 GB | 5 GB | 50 GB

basic questions

Input table: 1,000,000,000 rows x 9 columns (50 GB)

DuckDB	1.0.0	2024-07-04	25s
ClickHouse	24.5.1.1763	2024-06-07	28s
Polars	1.1.0	2024-07-09	47s
Datafusion	38.0.1	2024-06-07	56s
data.table	1.15.99	2024-06-07	88s
DataFrames.jl	1.6.1	2024-06-07	91s
InMemoryDatasets.jl	0.7.18	2023-10-17	218s
spark	3.5.1	2024-06-07	261s
R-arrow	16.1.0	2024-06-07	378s
collapse	2.0.14	2024-06-07	411s
(py)datatable	1.2.0a0	2024-06-07	1022s
dplyr	1.1.4	2024-06-07	1104s
pandas	2.2.2	2024-06-07	1126s
dask	2024.5.2	2024-06-07	out of memory
Modin		see README	pending

https://duckdblabs.github.io/db-benchmark/

Task

groupby | join

0.5 GB | 5 GB | 50 GB

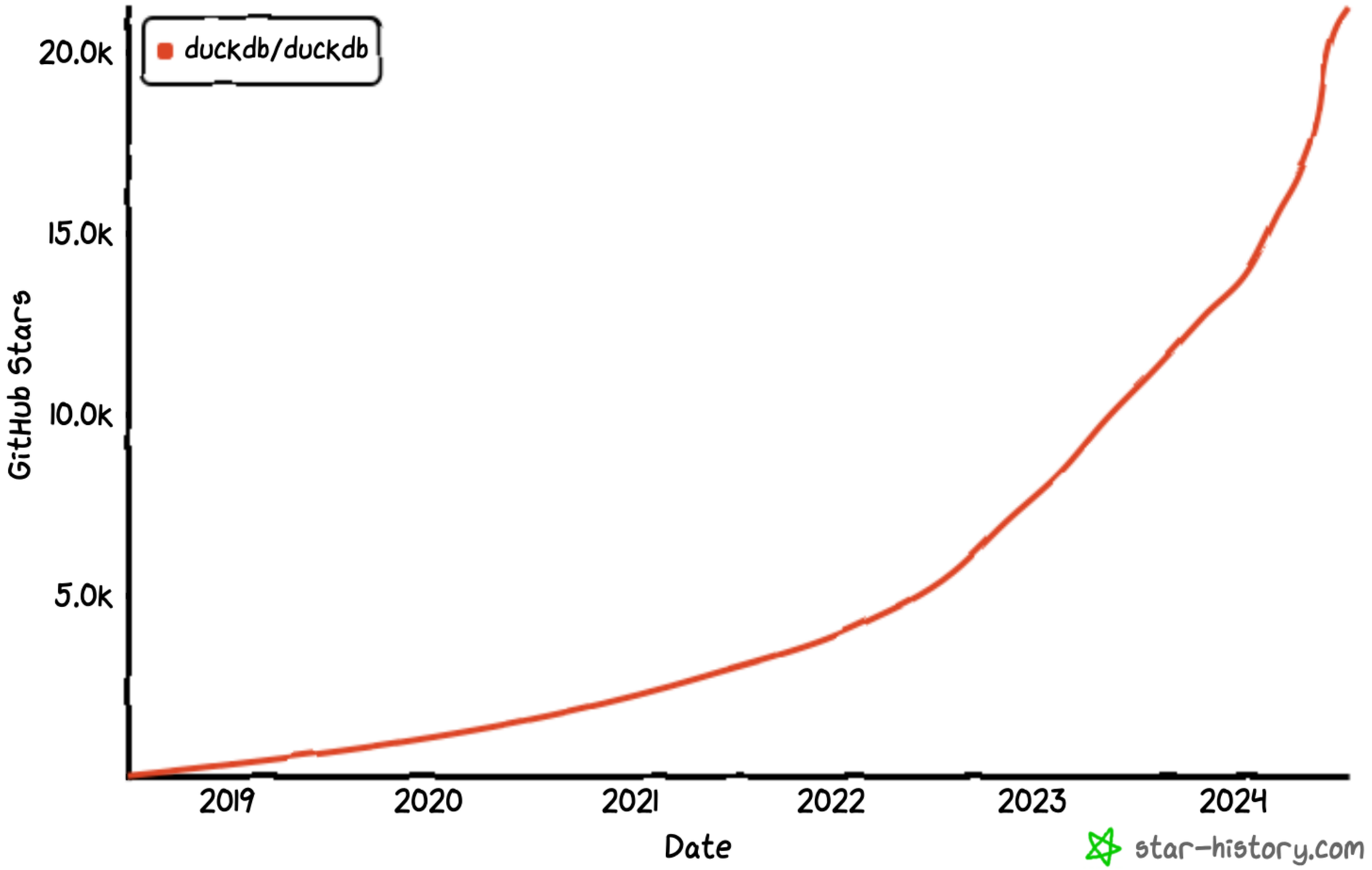
basic questions

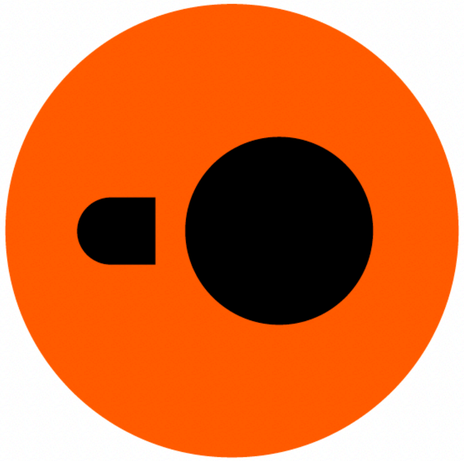
Input table: 1,000,000,000 rows x 7 columns (55 GB)

DuckDB*	1.0.0	2024-07-09	126s
Polars	1.1.0	2024-07-09	130s
InMemoryDatasets.jl	0.7.19	2024-06-12	232s
DataFrames.jl	1.6.1	2023-11-06	out of memory
data.table	1.14.9	2023-11-08	timeout
dplyr	1.1.3	2023-11-06	out of memory
pandas	2.1.1	2023-11-06	out of memory
spark*	3.5.0	2023-10-24	timeout
dask	2023.10.0	2023-11-29	out of memory
R-arrow	13.0.0.1	2023-11-06	out of memory
Datafusion	31.0.0	2023-10-25	undefined exception
ClickHouse	24.5.1.1763	2024-06-13	undefined exception
collapse	2.0.14	2024-06-11	undefined exception
Modin		see README	pending

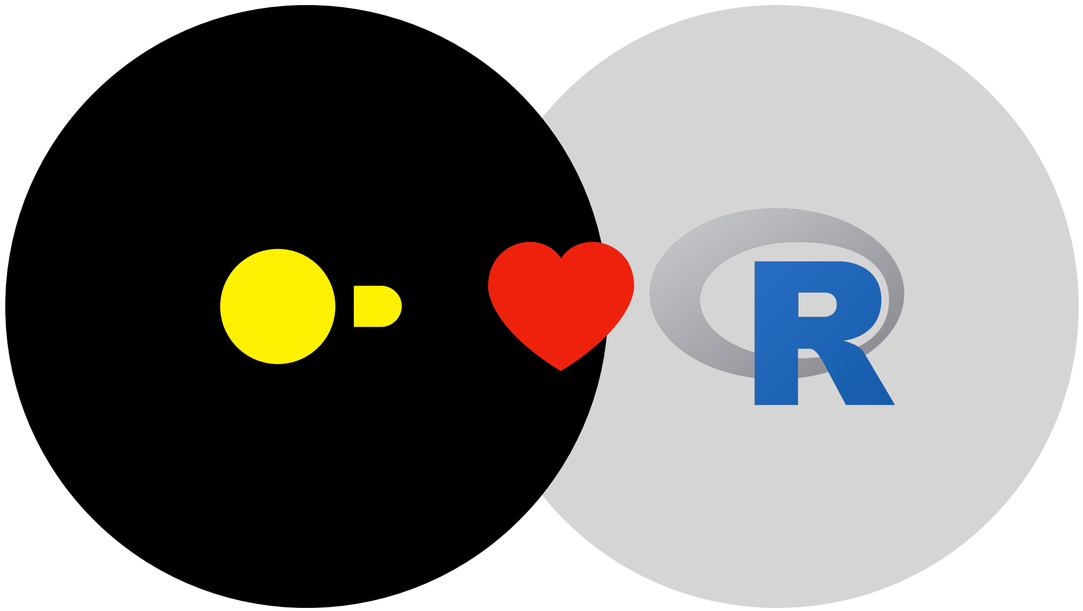
https://duckdblabs.github.io/db-benchmark/

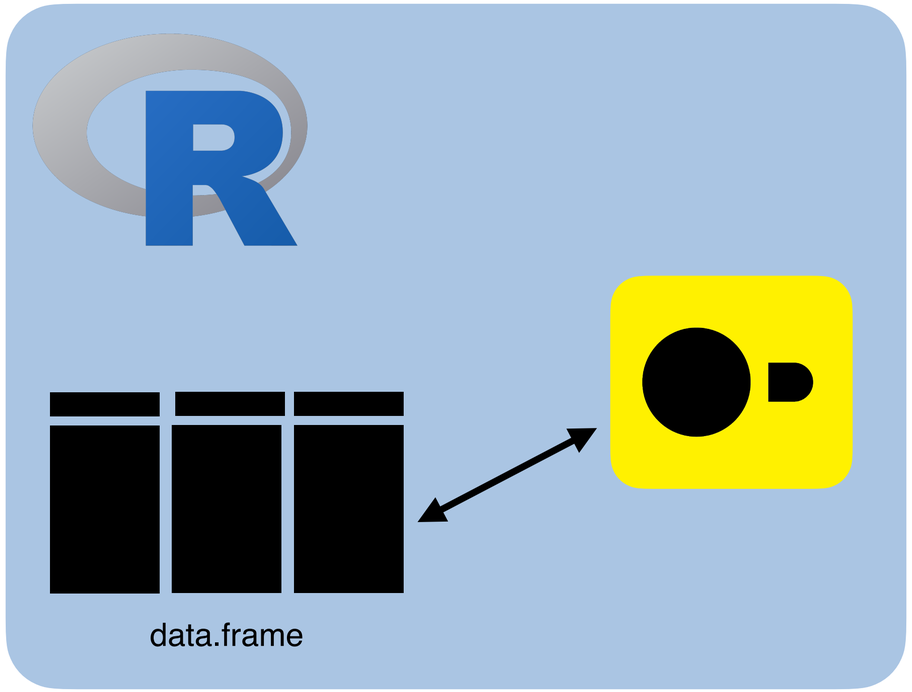
Free!

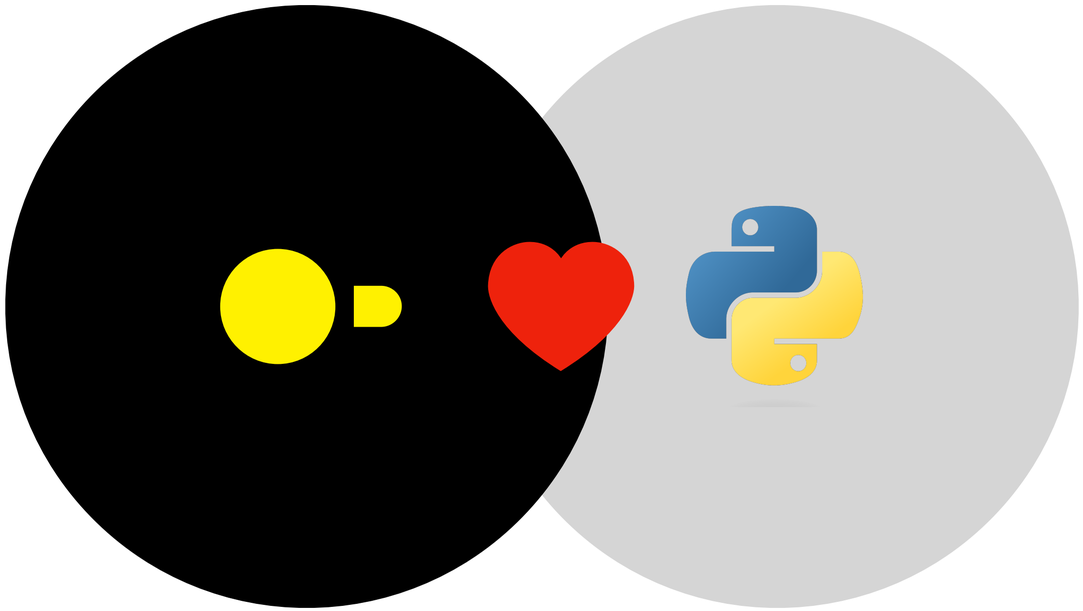

duckdb/duckdb
20.0k
15.0k
10.0k
5.0k
GitHub Stars
2019
2020
2021
2022
2023
2024
Date
star-history.com

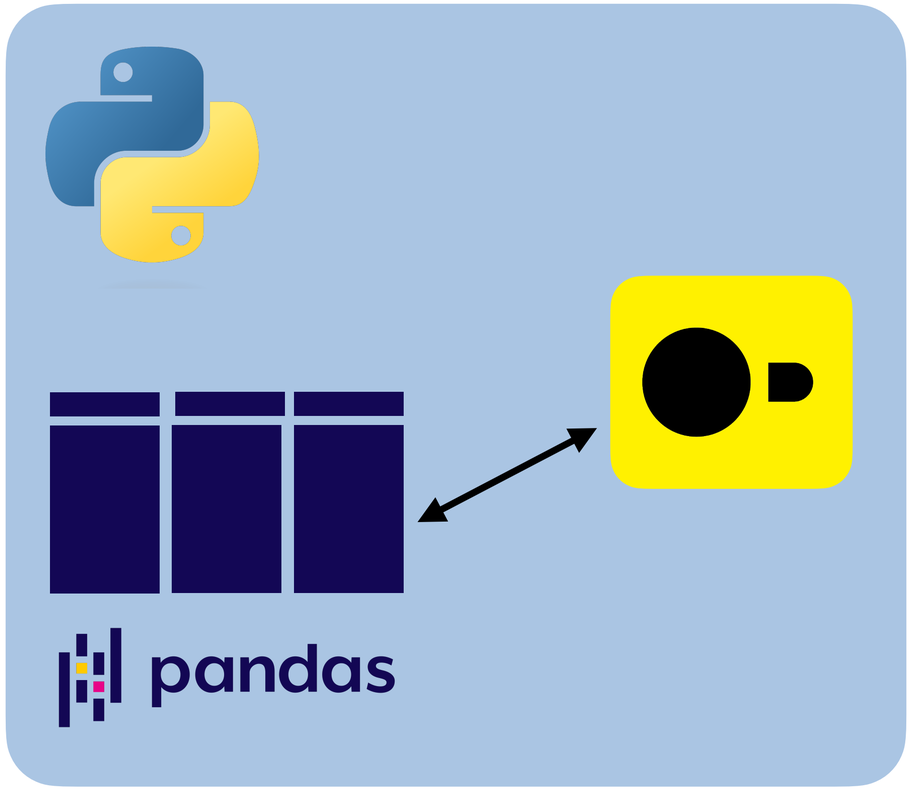


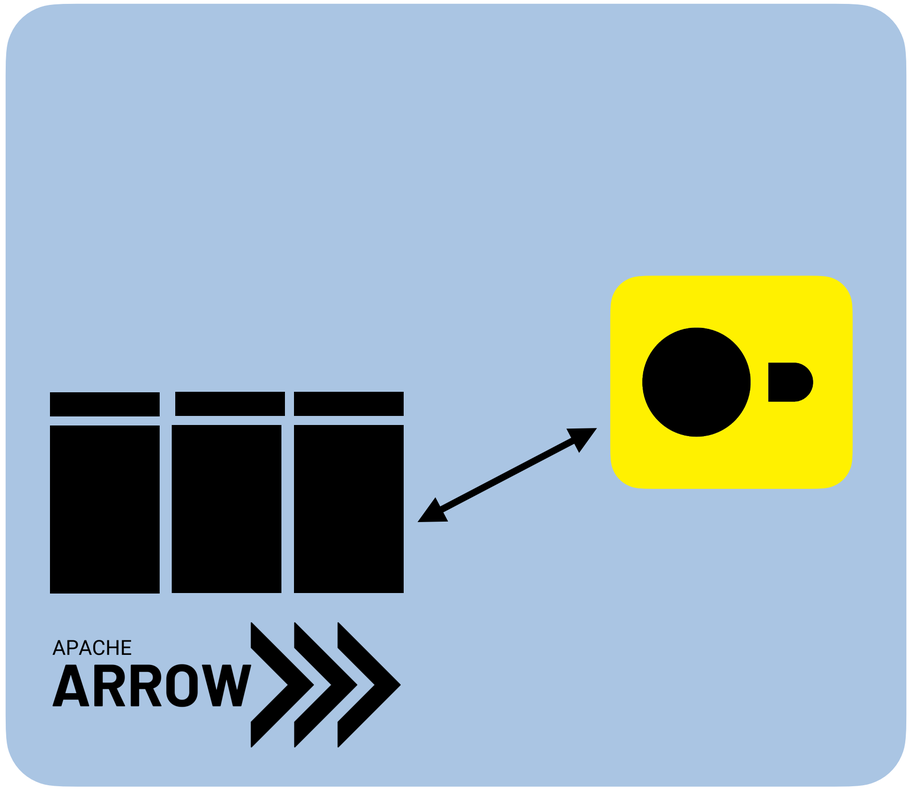
DuckDB Labs

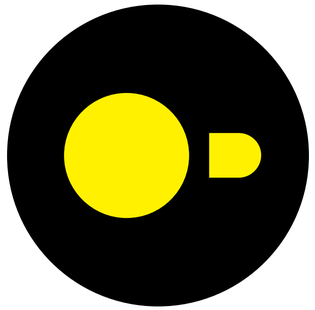



data.frame
R Process




pandas
Python Process


APACHE
ARROW
Python/R Process



DuckDB:
Slower and More Frustrating?

Multi-Hypothesis CSV Parsing

Till Döhmen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
tilldoehmen@gmail.com

Hannes Mühleisen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
hannes@cwi.nl

Peter Boncz
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
boncz@cwi.nl

ABSTRACT

Comma Separated Value (CSV) files are commonly used to represent data. CSV is a very simple format, yet we show that it gives rise to a surprisingly large amount of ambiguities in its parsing and interpretation. We summarize the state-of-the-art in CSV parsers, which typically make a linear series of parsing and interpretation decisions, such that any wrong decision at an earlier stage can negatively affect all downstream decisions. Since computation time

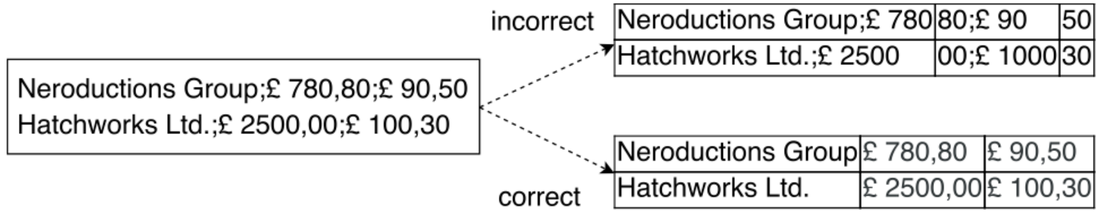


Figure 1: Ambiguous CSV file which is at risk to be parsed incorrectly, because the number of commas and the number of semi-colons per row are the same.

```
D FROM Data8277.csv;
```

Year int64	Age varchar	Ethnic int64	Sex int64	Area varchar	count varchar
2018	000	1	1	01	795
2018	000	1	1	02	5067
2018	000	1	1	03	2229
2018	000	1	1	04	1356
2018	000	1	1	05	180

…

```
D SUMMARIZE FROM "Data8277.csv";
```

column_name varchar	column_type varchar	min varchar	max varchar	approx_unique int64	va
Year	BIGINT	2006	2018	3	2012.3
Age	VARCHAR	000	999999	149	
Ethnic	BIGINT	1	9999	11	930.545
Sex	BIGINT	1	9	3	4.0
Area	VARCHAR	001	DHB9999	2048	
count	VARCHAR	..C	9999	16825	

System	Command	Time
R read.csv	read.csv("Data8277.csv")	22 s
PostgreSQL	COPY … FROM …	16 s
readr	readr::read_csv("Data8277.csv")	10 s
Pandas	pandas.read_csv("Data8277.csv")	5.5 s
data.table	fread("Data8277.csv")	2.9 s
Polars	polars.read_csv("Data8277.csv")	Exception
DuckDB	**CREATE TABLE … AS FROM Data8277.csv**	**1.1 s**

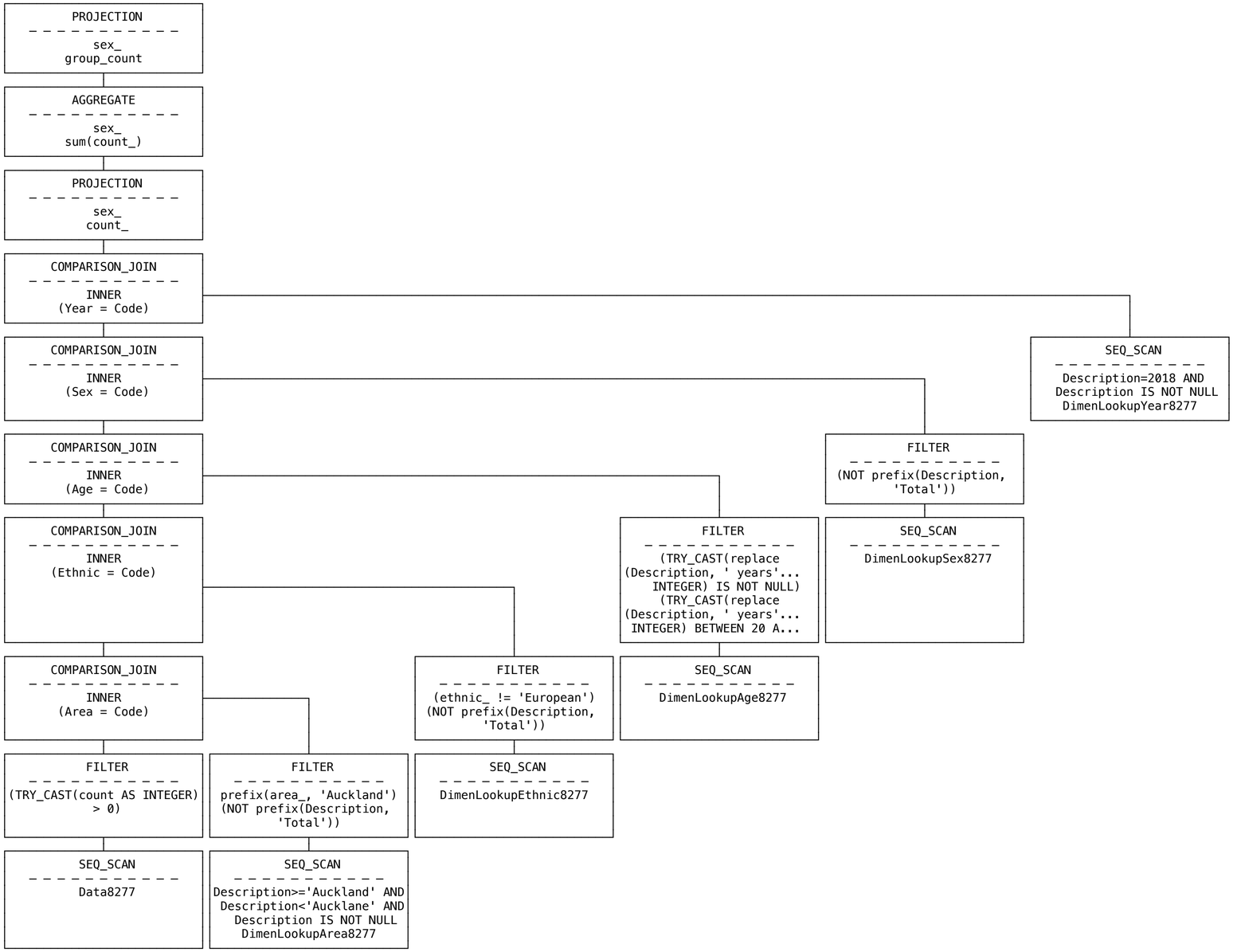
Complex Queries

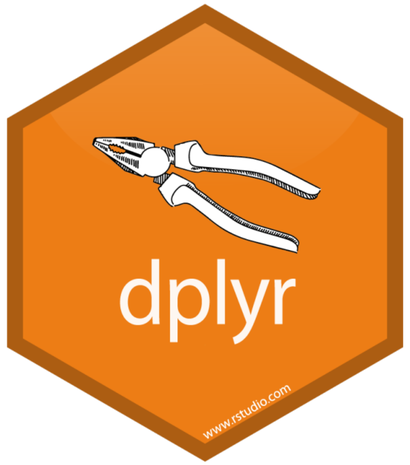
“Computer, give me the number of non-Europeans between 20 and 40 that live in the Auckland area from the 2018 census grouped by sex.”

```
WITH expanded_cleaned_data AS (
  SELECT year.Description year_,
    area.Description area_,
    ethnic.Description ethnic_,
    sex.Description sex_,
    try_cast(replace(age.Description, ' years', '') as integer) age_,
    try_cast(data.count as integer) count_
  FROM Data8277.csv data
    JOIN DimenLookupAge8277.csv age ON data.Age = age.Code
    JOIN DimenLookupArea8277.csv area ON data.Area = area.Code
    JOIN DimenLookupEthnic8277.csv ethnic ON data.Ethnic = ethnic.Code
    JOIN DimenLookupSex8277.csv sex ON data.Sex = sex.Code
    JOIN DimenLookupYear8277.csv year ON data.Year = year.Code
  WHERE
    count_ > 0
    AND age_ IS NOT NULL
    AND area_ NOT LIKE 'Total%'
    AND ethnic_ NOT LIKE 'Total%'
    AND sex_ NOT LIKE 'Total%'
)
SELECT sex_, SUM(count_) group_count
FROM expanded_cleaned_data
WHERE age_ BETWEEN 20 AND 40
AND area_ LIKE 'Auckland%'
AND ethnic_ <> 'European'
AND year_ = 2018
GROUP BY sex_;
```

1s

“Computer, give me the number of non-Europeans between 20 and 40 that live in the Auckland area from the 2018 census grouped by sex.”





“Computer, give me the number of non-Europeans between 20 and 40 that live in the Auckland area from the 2018 census grouped by sex.”

```
data   <- readr::read_csv("Data8277.csv")
…
expanded_cleaned_data <- data |>
  filter(grepl("^\\d+$", count)) |>
  mutate(count_ = as.integer(count)) |>
  filter(count_ > 0) |>
  inner_join(
    age |>
      filter(grepl("^\\d+ years$", Description)) |>
      mutate(age_ = as.integer(Code)),
    join_by(Age == Code)) |>
  inner_join(area |>
    mutate(area_ = Description) |>
    filter(!grepl("^Total", area_)), join_by(Area == Code)) |>
  inner_join(ethnic |>
    mutate(ethnic_ = Description) |>
    filter(!grepl("^Total", ethnic_)), join_by(Ethnic == Code)) |>
  inner_join(sex |>
    mutate(sex_ = Description) |>
    filter(!grepl("^Total", sex_)), join_by(Sex == Code)) |>
  inner_join(year |> mutate(year_ = Description) |>
    filter(year_ == "2018"), join_by(Year == Code))

twenty_till_fourty_non_european_in_auckland_area <-
  expanded_cleaned_data |>
  filter(
    age_ >= 20, age_ <= 40,
    grepl("^Auckland", area_),
    ethnic_ != "European") |>
  summarise(group_count = sum(count_), .by = sex_)
```

15s

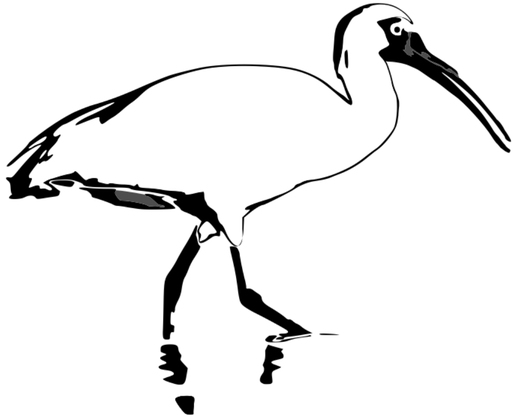
“Computer, give me the number of non-Europeans between 20 and 40 that live in the Auckland area from the 2018 census grouped by sex.”

```
data   <- duckplyr::duckplyr_df_from_csv("Data8277.csv")
…
expanded_cleaned_data <- data |>
  filter(grepl("^\\d+$", count)) |>
  mutate(count_ = as.integer(count)) |>
  filter(count_ > 0) |>
  inner_join(
    age |>
      filter(grepl("^\\d+ years$", Description)) |>
      mutate(age_ = as.integer(Code)),
    join_by(Age == Code)) |>
  inner_join(area |>
    mutate(area_ = Description) |>
    filter(!grepl("^Total", area_)), join_by(Area == Code)) |>
  inner_join(ethnic |>
    mutate(ethnic_ = Description) |>
    filter(!grepl("^Total", ethnic_)), join_by(Ethnic == Code)) |>
  inner_join(sex |>
    mutate(sex_ = Description) |>
    filter(!grepl("^Total", sex_)), join_by(Sex == Code)) |>
  inner_join(year |> mutate(year_ = Description) |>
    filter(year_ == "2018"), join_by(Year == Code))

twenty_till_fourty_non_european_in_auckland_area <-
  expanded_cleaned_data |>
  filter(
    age_ >= 20, age_ <= 40,
    grepl("^Auckland", area_),
    ethnic_ != "European") |>
  summarise(group_count = sum(count_), .by = sex_)
```

1.5s


```
> data   <- readr::read_csv("Data8277.csv")
> data2  <- data |> filter(Area=='01')
> result <- data2 |> summarize(sum(as.integer(count), na.rm=T))
```

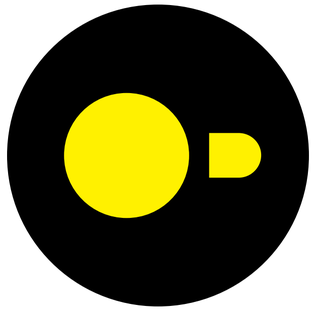
```
install.packages("duckplyr")
```



“Computer, give me the number of non-Europeans between 20 and 40 that live in the Auckland area from the 2018 census grouped by sex.”

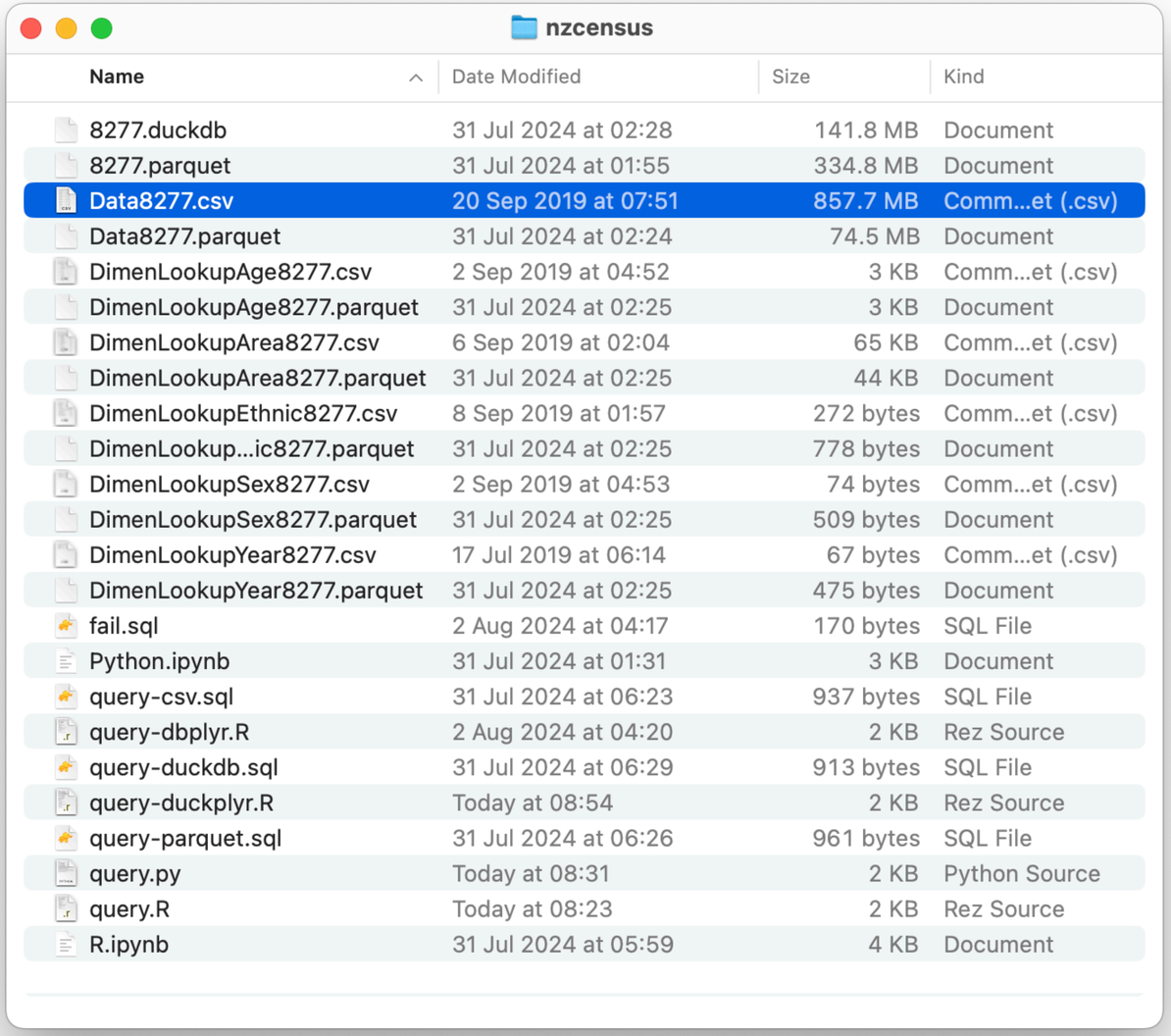
```
con = ibis.connect("duckdb://")
data   = con.read_csv("Data8277.csv")
…

data_ = data.filter(_["count"].re_search("^\\d+$")).mutate(count_=_["count"].cast("int32")).drop(_["count"])
age_ = age.filter(_.Description.re_search("^\\d+ years$")).mutate(age_code=_.Code, age_value=_.Description.r
years", "").cast("int32")).drop([_.SortOrder, _.Description, _.Code])
area_ = area.mutate(area_code=_.Code, area_value=_.Description).filter(-
_.area_value.re_search("^Total")).drop([_.SortOrder, _.Description, _.Code])
ethnic_ = ethnic.mutate(ethnic_code=_.Code, ethnic_value=_.Description).filter(-
_.ethnic_value.re_search("^Total")).drop([_.SortOrder, _.Description, _.Code])
sex_ = sex.mutate(sex_code=_.Code, sex_value = _.Description).filter(-
_.sex_value.re_search("^Total")).drop([_.SortOrder, _.Description, _.Code])
year_ = year.mutate(year_code=_.Code, year_value=_.Description).filter(_.year_value == 2018).drop([_.SortOrd
_.Description, _.Code])

expanded_cleaned_data = data_.join(age_, _.Age == age_.age_code).join(area_, _.Area ==
area_.area_code).join(ethnic_, _.Ethnic == ethnic_.ethnic_code).join(sex_, _.Sex == sex_.sex_code).join(year_,
== year_.year_code)

twenty_till_fourty_non_european_in_auckland_area = expanded_cleaned_data.filter([_.age_value >= 20, _.age_
40, _.area_value.re_search("^Auckland"), _.ethnic_value !=
"European"]).group_by(_.sex_value).aggregate(group_count=_.count_.sum())
```

1.3s



Atomic Datasets with DuckDB

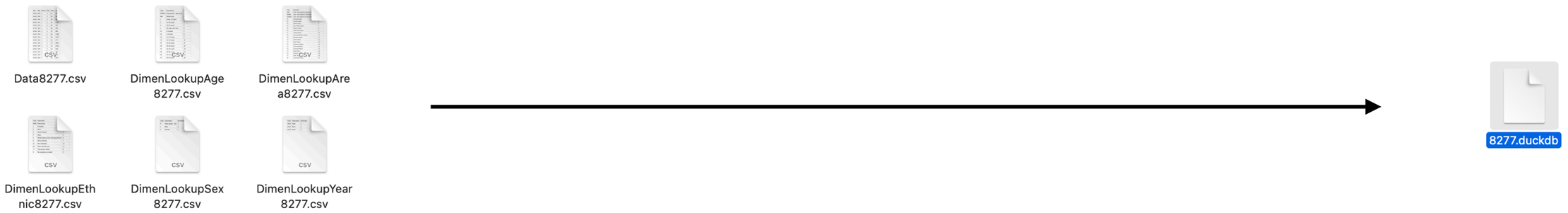

```
COPY Data8277.csv
TO   Data8277.parquet;
```


nzcensus
Name
Date Modified
Size
Kind
8277.duckdb 31 Jul 2024 at 02:28 141.8 MB Document
8277.parquet 31 Jul 2024 at 01:55 334.8 MB Document
Data8277.csv 20 Sep 2019 at 07:51 857.7 MB Comm…et (.csv)
Data8277.parquet 31 Jul 2024 at 02:24 74.5 MB Document
DimenLookupAge8277.csv 2 Sep 2019 at 04:52 3 KB Comm…et (.csv)
DimenLookupAge8277.parquet 31 Jul 2024 at 02:25 3 KB Document
DimenLookupArea8277.csv 6 Sep 2019 at 02:04 65 KB Comm…et (.csv)
DimenLookupArea8277.parquet 31 Jul 2024 at 02:25 44 KB Document
DimenLookupEthnic8277.csv 8 Sep 2019 at 01:57 272 bytes Comm…et (.csv)
DimenLookup…ic8277.parquet 31 Jul 2024 at 02:25 778 bytes Document
DimenLookupSex8277.csv 2 Sep 2019 at 04:53 74 bytes Comm…et (.csv)
DimenLookupSex8277.parquet 31 Jul 2024 at 02:25 509 bytes Document
DimenLookupYear8277.csv 17 Jul 2019 at 06:14 67 bytes Comm…et (.csv)
DimenLookupYear8277.parquet 31 Jul 2024 at 02:25 475 bytes Document
fail.sql 2 Aug 2024 at 04:17 170 bytes SQL File
Python.ipynb 31 Jul 2024 at 01:31 3 KB Document
query-csv.sql 31 Jul 2024 at 06:23 937 bytes SQL File
query-dbplyr.R 2 Aug 2024 at 04:20 2 KB Rez Source
query-duckdb.sql 31 Jul 2024 at 06:29 913 bytes SQL File
query-duckplyr.R Today at 08:54 2 KB Rez Source
query-parquet.sql 31 Jul 2024 at 06:26 961 bytes SQL File
query.py Today at 08:31 2 KB Python Source
query.R Today at 08:23 2 KB Rez Source
R.ipynb 31 Jul 2024 at 05:59 4 KB Document

BEGIN TRANSACTION;

CREATE TABLE Data8277 AS FROM Data8277.csv;
CREATE TABLE DimenLookupAge8277 AS FROM DimenLookupAge8277.csv;
...

COMMIT;



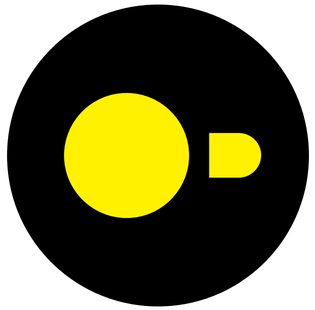
```
CREATE VIEW expanded_cleaned_data AS
    SELECT year.Description year_,
    area.Description area_,
    ethnic.Description ethnic_,
    sex.Description sex_,
    try_cast(replace(age.Description, ' years', '') as integer) age_,
    try_cast(data.count as integer) count_
  FROM Data8277 data
    JOIN DimenLookupAge8277 age ON data.Age = age.Code
    JOIN DimenLookupArea8277 area ON data.Area = area.Code
    JOIN DimenLookupEthnic8277 ethnic ON data.Ethnic = ethnic.Code
    JOIN DimenLookupSex8277 sex ON data.Sex = sex.Code
    JOIN DimenLookupYear8277 year ON data.Year = year.Code
  WHERE
    count_ > 0
    AND age_ IS NOT NULL
    AND area_ NOT LIKE 'Total%'
    AND ethnic_ NOT LIKE 'Total%'
    AND sex_ NOT LIKE 'Total%';
```

```
SELECT
…
WHERE area_   NOT LIKE 'Total%'
  AND ethnic_ NOT LIKE 'Total%'
  AND sex_    NOT LIKE 'Total%';
```

```
CREATE MACRO NOT_TOTAL(COL) AS
   COL NOT LIKE 'Total%';

SELECT
…
WHERE NOT_TOTAL(area_)
  AND NOT_TOTAL(ethnic_)
  AND NOT_TOTAL(sex_);
```

```
BEGIN TRANSACTION;
CREATE TABLE age_count (Age VARCHAR, COUNT INTEGER);
INSERT INTO age_count
    SELECT Age, CAST(count as INTEGER) count_ FROM Data8277;
Conversion Error: Could not convert string '..C' to INT32
LINE 2:  SELECT Age, CAST(count as INT…
-- oops

ROLLBACK;
-- nothing happened, table age_count does not exist
```

Just put your data in DuckDB:
Not doing so will only be **slower**
and more **frustrating.**



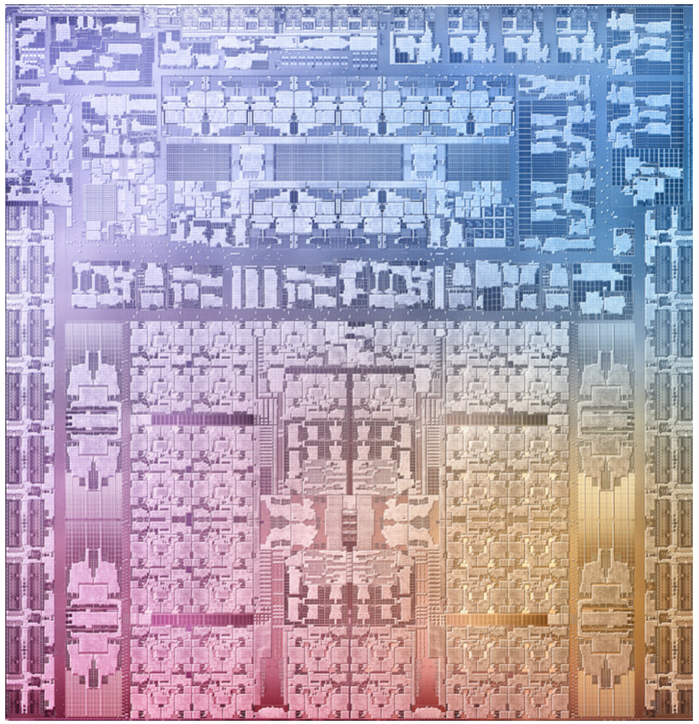
Big Data Is Dead?

Human typing speed	200 Characters/min
NL working population	10 000 000
NL work year	100 000 Minutes
Total data produced	200 000 000 000 000 Bytes
	200 Terabytes
After compression	60 Terabytes

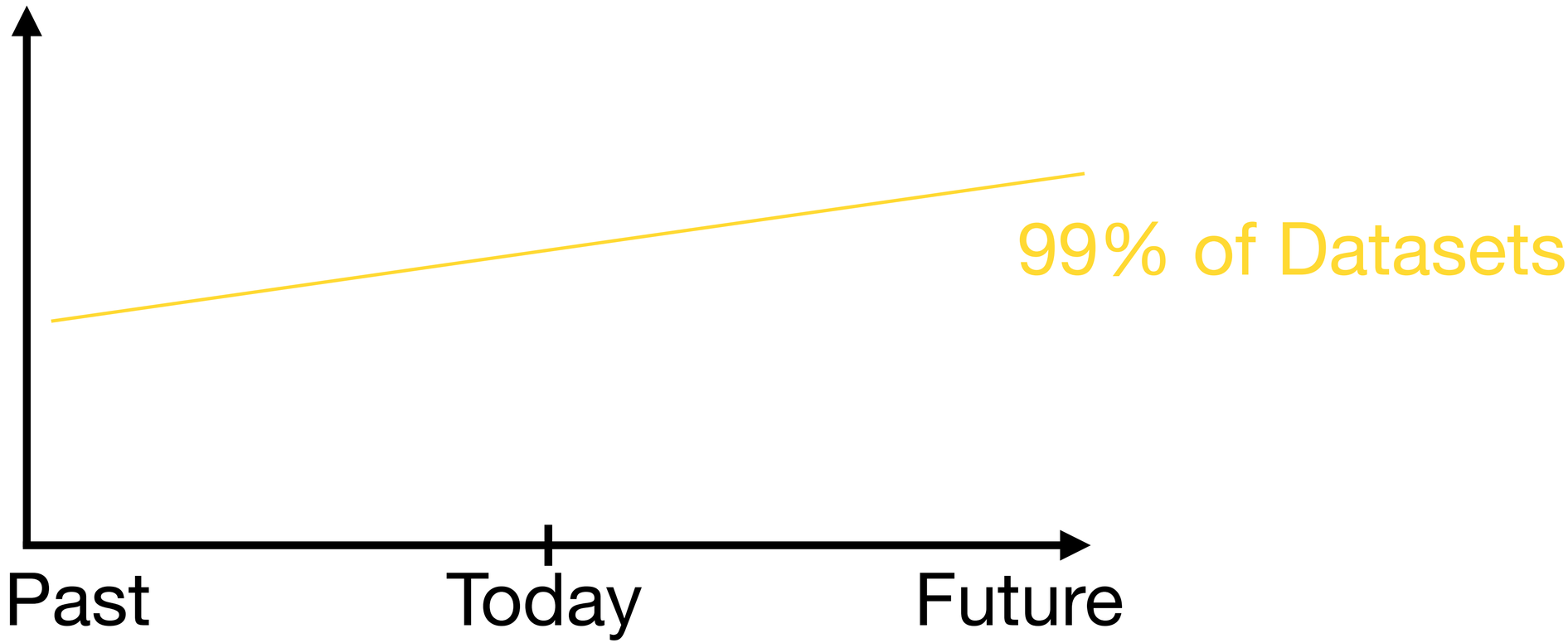

SEAGATE
mozaic 3+
EXOS™

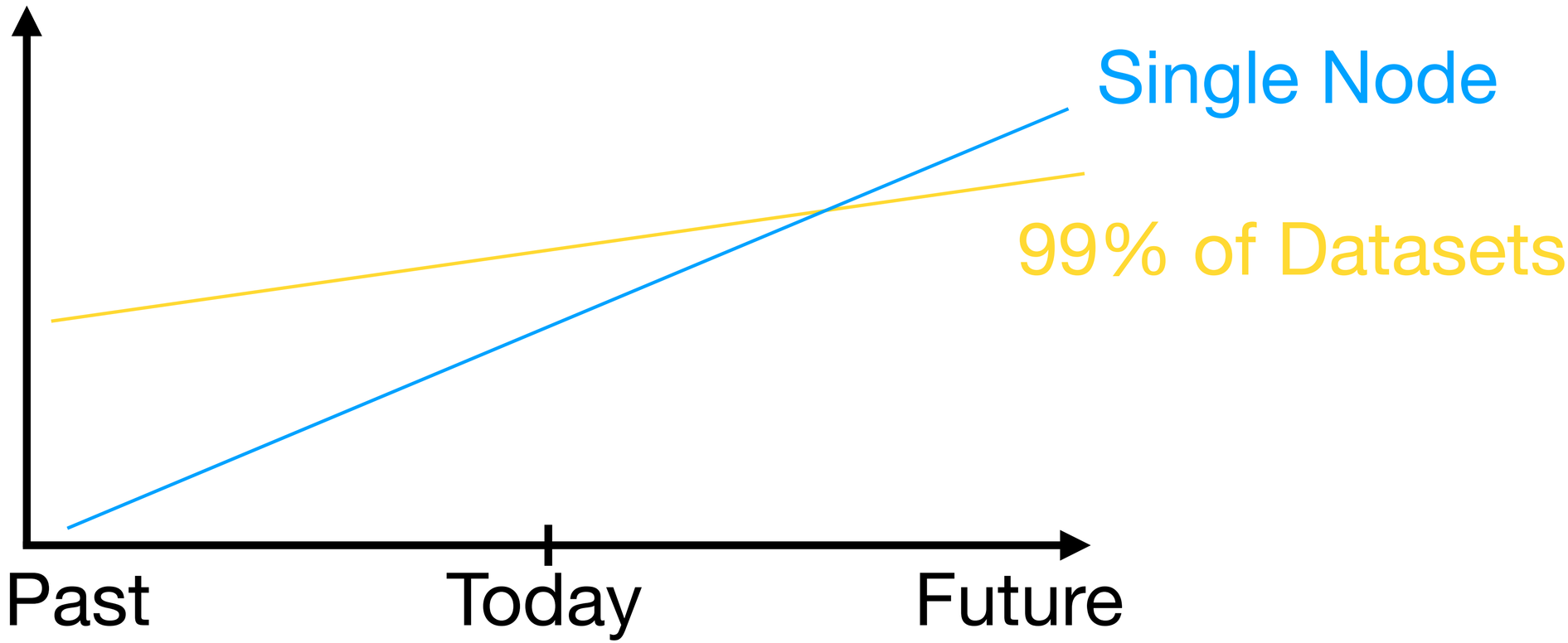

SEAGATE
mozaic 3+
EXOS™

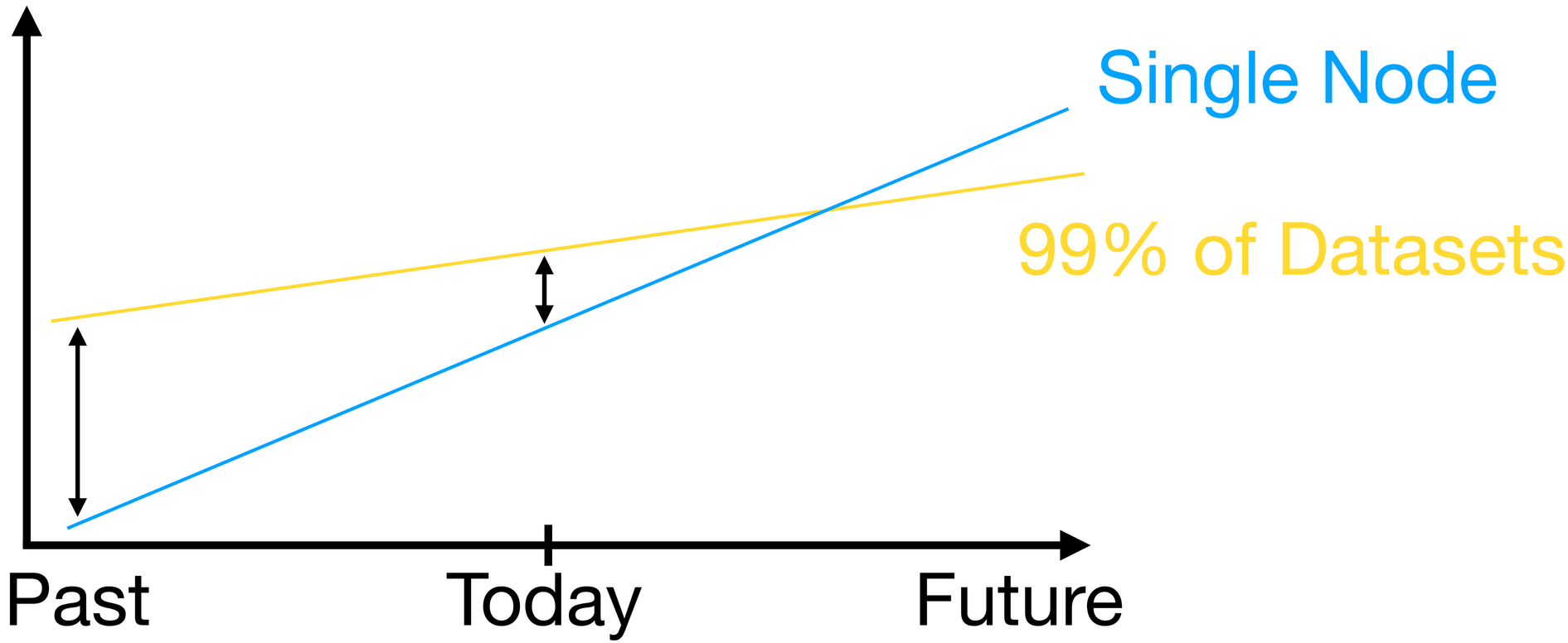
CPUs

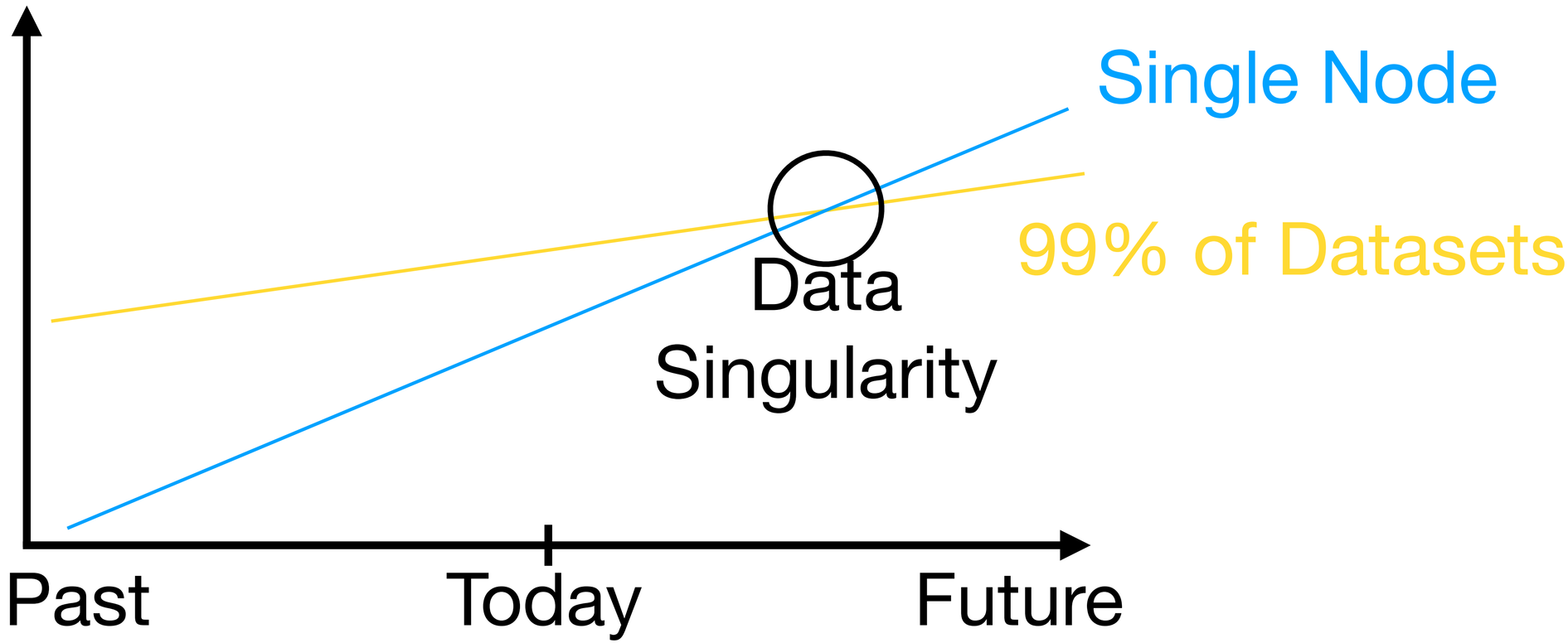


SSDs


99% of Datasets
Past
Today
Future


Single Node
99% of Datasets
Past
Today
Future


Single Node
99% of Datasets
Past
Today
Future


Single Node
99% of Datasets
Data
Singularity
Past
Today
Future

Scalability! But at what COST?

Frank McSherry
Unaffiliated

Michael Isard
Unaffiliated*

Derek G. Murray
Unaffiliated†

Abstract

We offer a new metric for big data platforms, COST, or the Configuration that Outperforms a Single Thread. The COST of a given platform for a given problem is the hardware configuration required before the platform outperforms a competent single-threaded implementation. COST weighs a system's scalability against the overheads introduced by the system, and indicates the actual performance gains of the system, without rewarding systems that bring substantial but parallelizable overheads.

We survey measurements of data-parallel systems recently reported in SOSP and OSDI, and find that many systems have either a surprisingly large COST, often

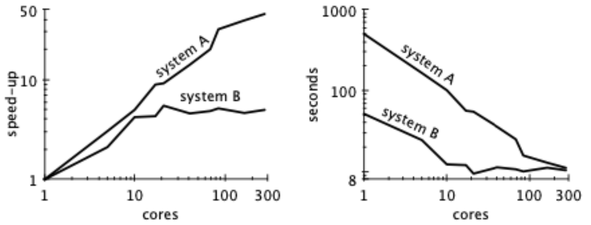


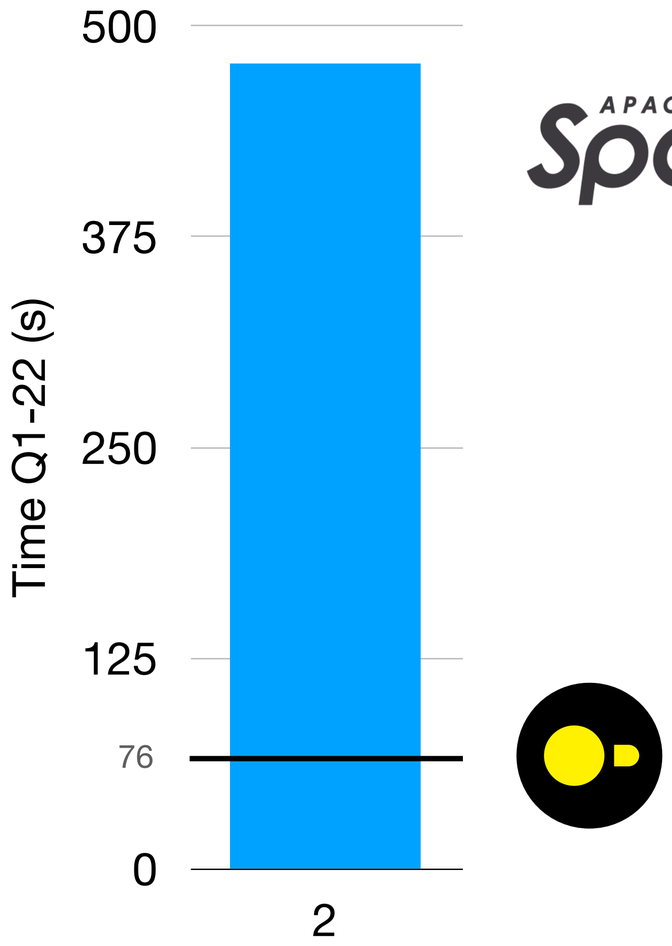
Figure 1: Scaling and performance measurements for a data-parallel algorithm, before (system A) and after (system B) a simple performance optimization. The unoptimized implementation "scales" far better, despite (or rather, because of) its poor performance.

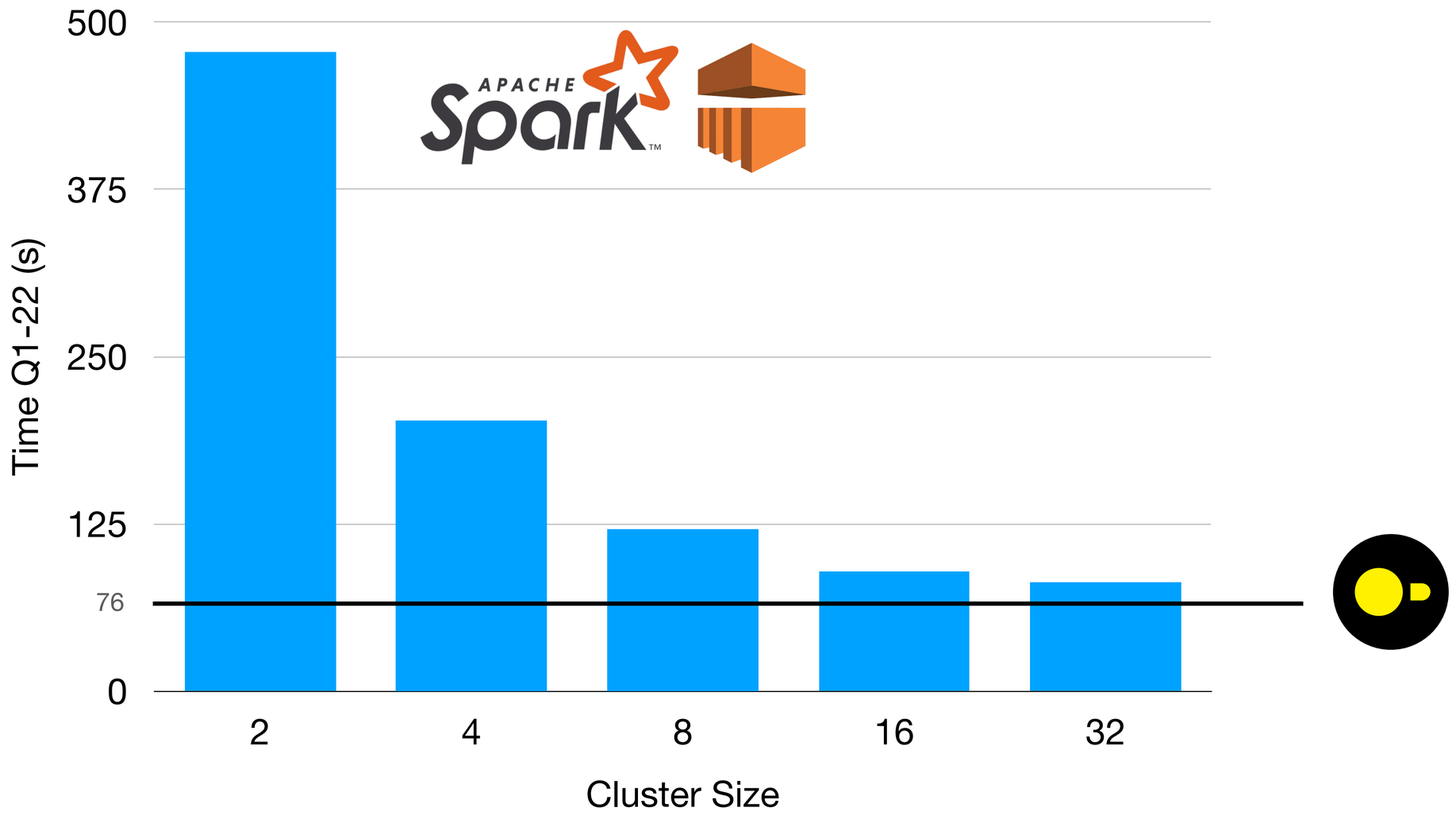
2024/08/07 - Jordan Tigani

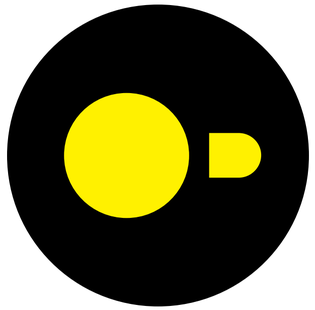
REDSHIFT FILES: THE HUNT FOR BIG DATA

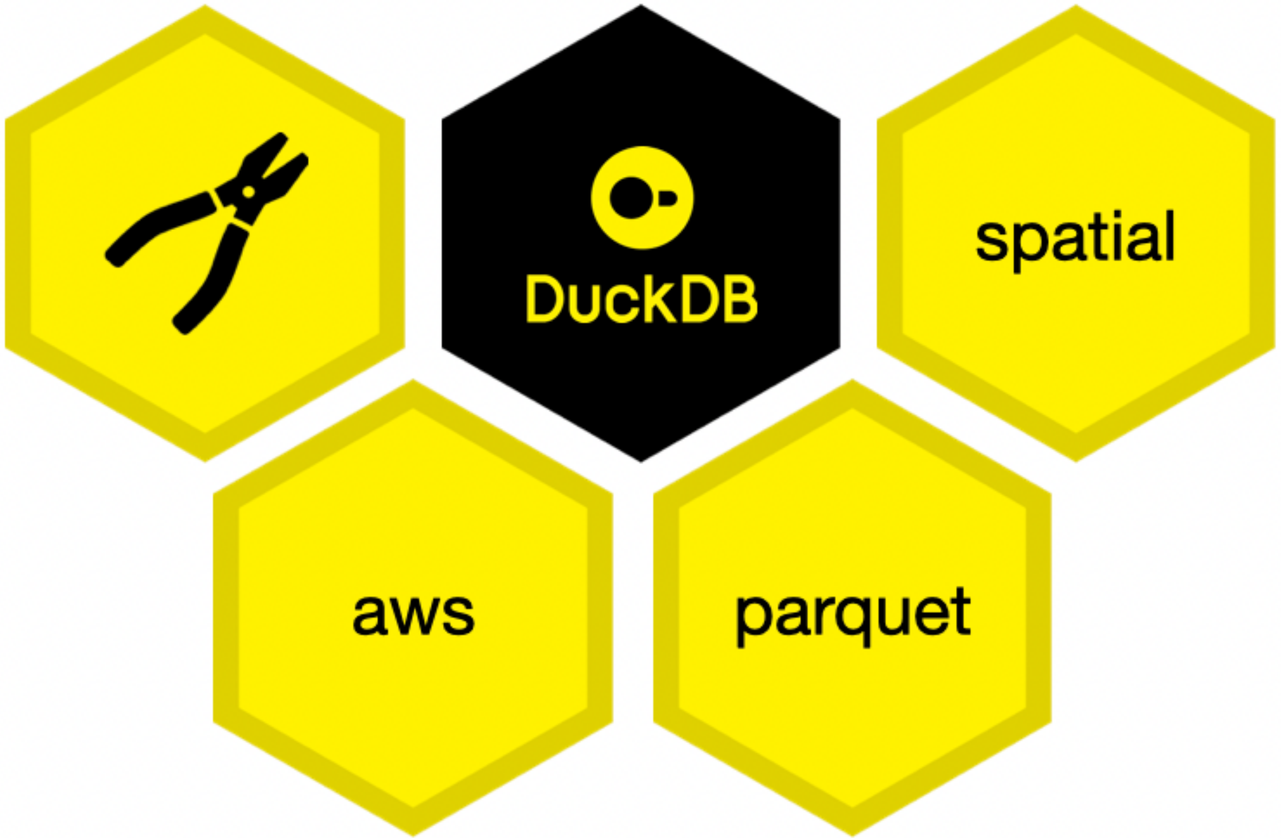
Jordan Tigani revisits his popular Big Data is Dead blog post with analysis of the data from the Redshift TPC is Not Enough paper.

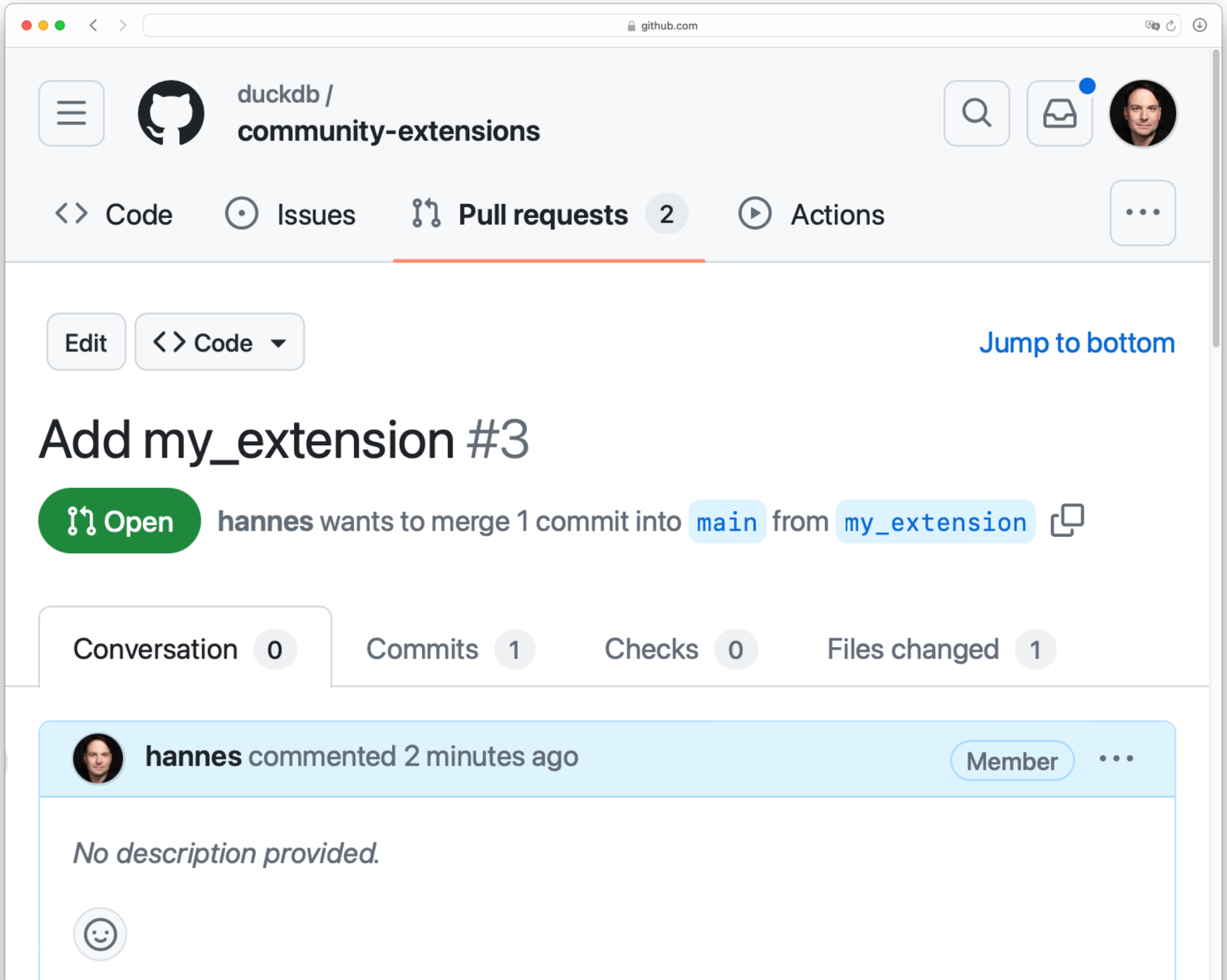
Experiments!


500
375
250
125
76
0
Time Q1-22 (s)
2


500
375
250
125
76
0
Time Q1-22 (s)
2
4
8
16
32
Cluster Size




DuckDB
spatial
aws
parquet


github.com
duckdb /
community-extensions
Code
Issues
Pull requests 2
Actions
Edit
Code
Jump to bottom
Add my_extension #3
Open
hannes wants to merge 1 commit into main from my_extension
Conversation 0
Commits 1
Checks 0
Files changed 1
hannes commented 2 minutes ago
Member
No description provided.

INSTALL my_extension FROM community;
LOAD my_extension;

Data Wrangling Like a Boss With DuckDB

- Most databases are frustrating and slow
  - Not DuckDB!
- More cool database features
- Single node is the future
- Big Data is kind of dead

hannes.muehleisen.org
@hfmuehleisen

DuckCon #5
Seattle, WA
2024-08-15

DuckDB Foundation