Exploration of DuckDB

JORRIT VANDER MYNSBRUGGE

GROUP 14

SID 0606134

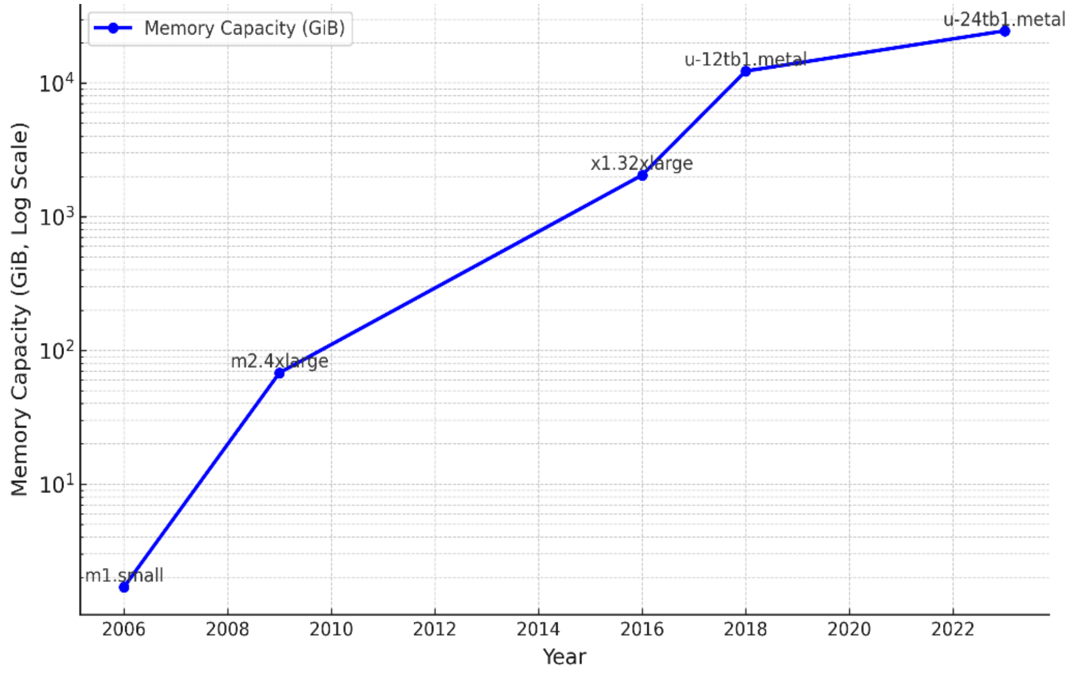
WORKING STUDENT

Context

Exponential growth of hardware

Amazon EC2 offering

- 2006 - m1.small – 1 CPU + 1.7GB RAM
- 2024 - x8g.metal-48xl – 192vCPU + 3TB RAM
- 2024 - High-Memory (U-1) – 448vCPU + 24TB RAM



Rise of ‘small data’ movement

Big Data

Consensus,
Paxos,
Raft

Data alignment

Small Data

Node failures

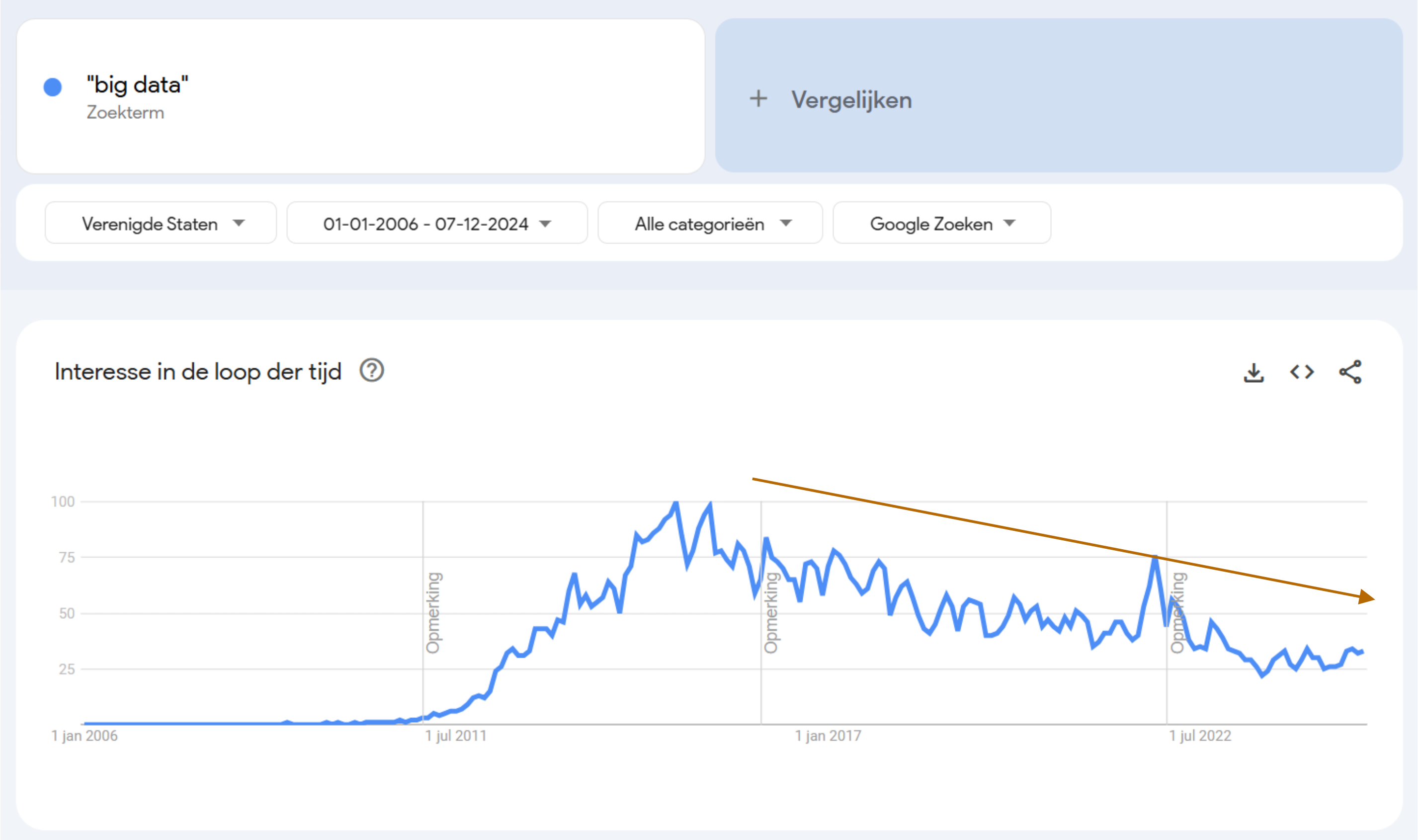
Stale data
relevance?

Data shuffle

Eventual consistency

CAP theorem

Google trends “big data”



Google trends “small data”

The DuckDB sales pitch

"DuckDB: The SQLite for Analytics – Power, Simplicity, and Portability.“

In proces DB

Vectorized Query Engine

ETL / OLAP

Simplicity

Ease of use

Small data

Many formats

ACID

Many bindings

	Transactional	Analytical
Embedded	SQLite, SolidDB	DuckDB
Stand-alone	Postgres, MySQL	Snowflake, ClickHouse, Redshift

Experiments

Research questions

OLTP vs OLAP using TPC-DS

- How does DuckDB perform against a traditional RDBMS under analytical workload conditions?

Scalability using TPC-H

- How does DuckDB scale with growing database size?

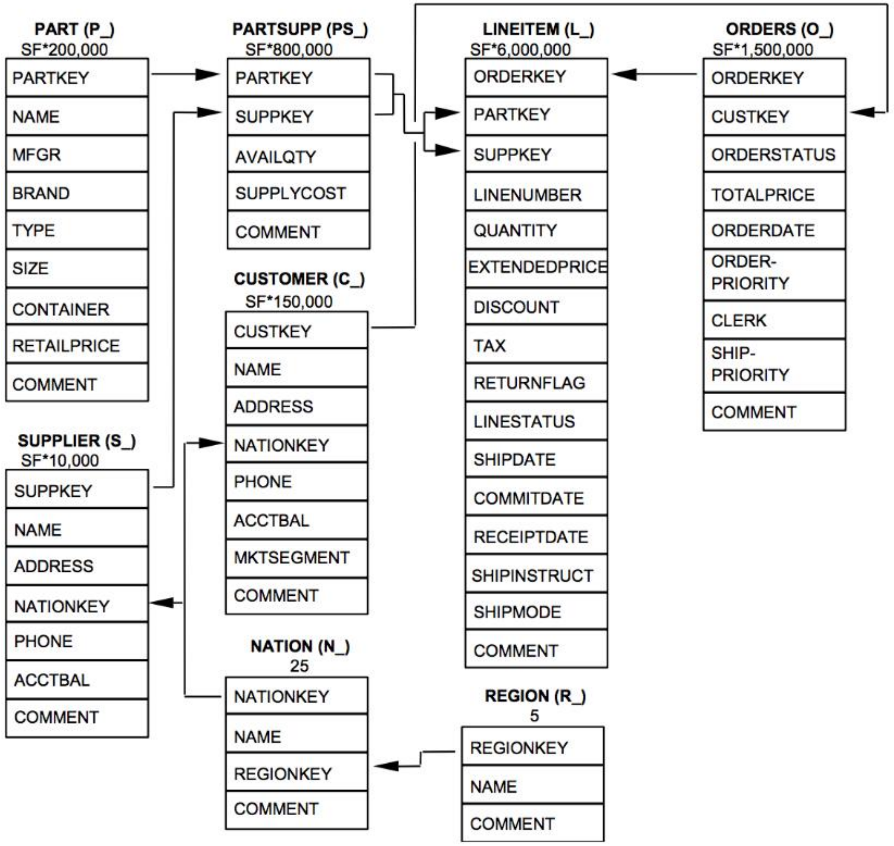
Comparative analysis using db-benchmark

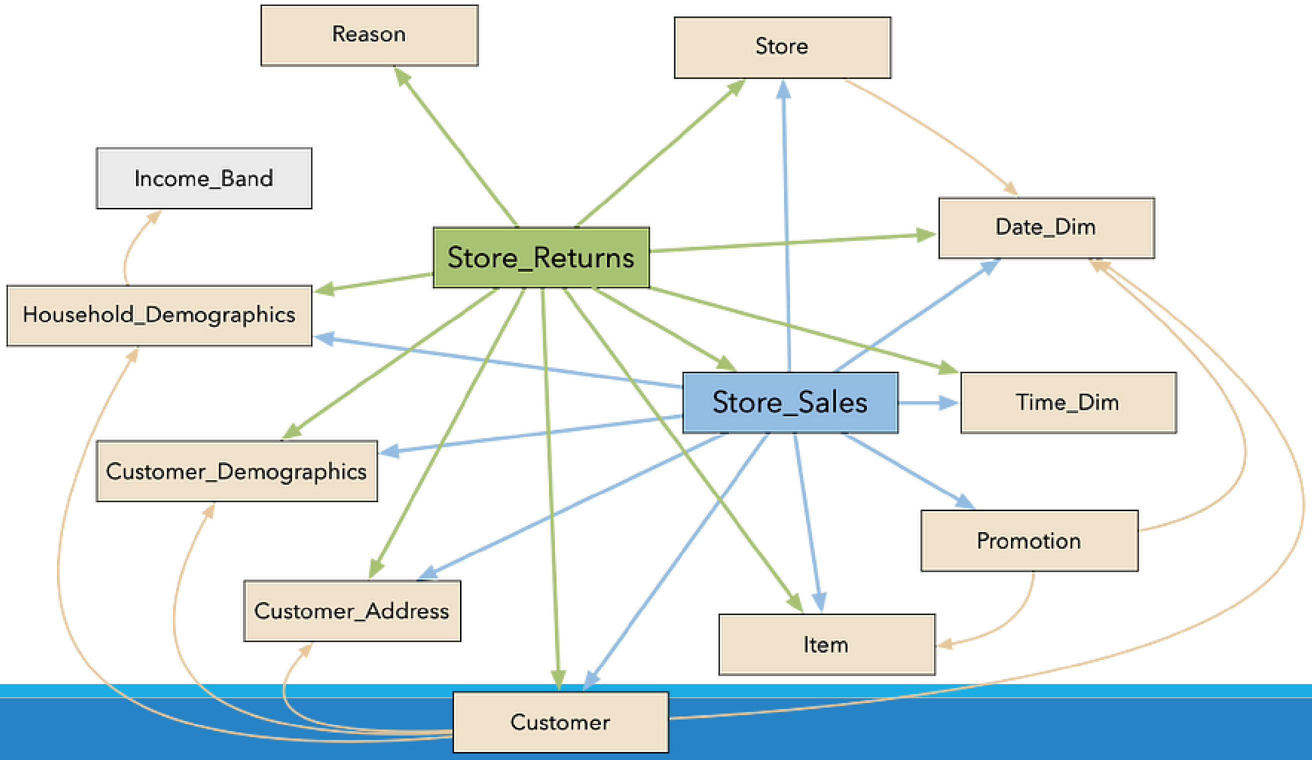
- How does DuckDB hold up against other in-memory OLAP processing systems?

source:
https://docs.snowflake.com/en/user-guide/sample-data-tpch
https://medium.com/hyrise/a-summary-of-tpc-ds-9fb5e7339a35

TPC-H vs TPC-DS

	TPC-H	TPC-DS
Schema type	3rd Normal Form	Multiple Snowflake
Number of tables	8	24
Number of columns (min)	3	3
Number of columns (max)	16	34
Number of columns (avg)	~ 7.6	18
Number of foreign keys	9	104





OLTP vs OLAP

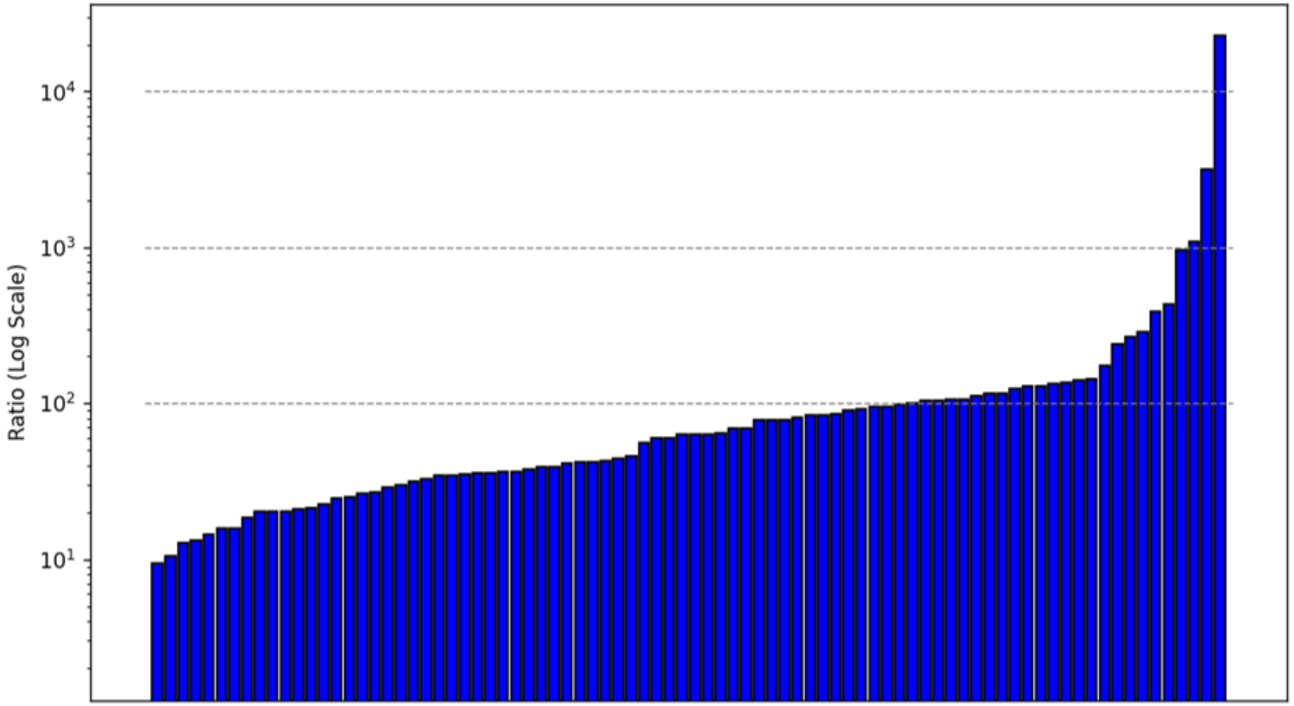


Figure 3 Speedup of DuckDB over Postgress for the 84 TPC-DS queries that did not time out on postgres.

Query	Postgres [s]	DuckDB [s]	Ratio
1	timeout	0,097	
2	3,748	0,141	27
3	6,891	0,082	84
4	timeout	1,589	
5	4,945	0,201	25
6	timeout	0,162	
7	4,175	0,117	36
8	1,358	0,067	20
9	13,401	0,369	36
10	64,713	0,067	966
11	timeout	0,981	
12	0,536	0,042	13
13	3,056	0,122	25
14	timeout	1,356	
15	1,176	0,052	23
16	timeout	0,031	
17	timeout	0,099	
18	4,094	0,190	22
19	3,271	0,074	44
20	2,224	0,053	42
21	10,645	0,037	288
22	158,019	1,838	86
23	156,759	1,112	141
24	5,430	0,171	32
25	timeout	0,077	
26	4,080	0,096	42
27	4,674	0,250	19
28	28,235	0,295	96
29	11,100	0,142	78
30	timeout	0,123	
31	43,010	0,162	265
32	250,402	0,011	22764
33	6,754	0,074	91

Query	Postgres [s]	DuckDB [s]	Ratio
34	4,232	0,101	42
35	timeout	0,305	
36	8,791	0,250	35
37	18,750	0,043	436
38	45,684	0,262	174
39	108,574	0,280	388
40	3,121	0,056	56
41	33,772	0,031	1089
42	4,282	0,041	104
43	8,448	0,089	95
44	16,202	0,118	137
45	1,506	0,095	16
46	6,077	0,168	36
47	58,687	0,528	111
48	12,484	0,149	84
49	17,403	0,129	135
50	8,067	0,175	46
51	33,327	2,090	16
52	3,868	0,049	79
53	4,718	0,078	60
54	timeout	0,103	
55	3,911	0,048	81
56	8,690	0,111	78
57	24,801	0,359	69
58	9,338	0,090	104
59	37,343	0,289	129
60	8,755	0,127	69
61	7,974	0,125	64
62	6,321	0,063	100
63	4,446	0,069	64
64	18,913	0,545	35
65	26,490	0,415	64
66	5,969	0,183	33

Query	Postgres [s]	DuckDB [s]	Ratio
67	85,954	4,063	21
68	4,034	0,200	20
69	17,325	0,149	116
70	22,054	0,189	117
71	7,351	0,246	30
72	44,301	0,414	107
73	3,864	0,106	36
74	timeout	0,842	
75	28,367	0,446	64
76	9,813	0,162	61
77	9,212	0,093	99
78	26,832	1,856	14
79	4,845	0,180	27
80	11,426	0,328	35
81	timeout	0,262	
82	19,376	0,080	242
83	1,169	0,089	13
84	0,929	0,099	9
85	5,123	0,124	41
86	4,751	0,121	39
87	47,340	0,379	125
88	32,602	0,359	91
89	6,456	0,164	39
90	2,888	0,027	107
91	0,735	0,036	20
92	148,510	0,047	3160
93	7,959	0,275	29
94	timeout	0,089	
95	timeout	0,730	
96	4,541	0,035	130
97	14,309	0,381	38
98	5,216	0,494	11
99	14,206	0,099	143

Scalability

File name	Scale Factor (GB uncompressed)	Size on disk in MB (compressed)
tpch-sf1.db	1	254
tpch-sf3.db	3	771
tpch-sf10.db	10	2.612
tpch-sf30.db	30	7.965
tpch-sf100.db	100	26.962

Table 1 Overview of the TPC-H databases used for the benchmark.

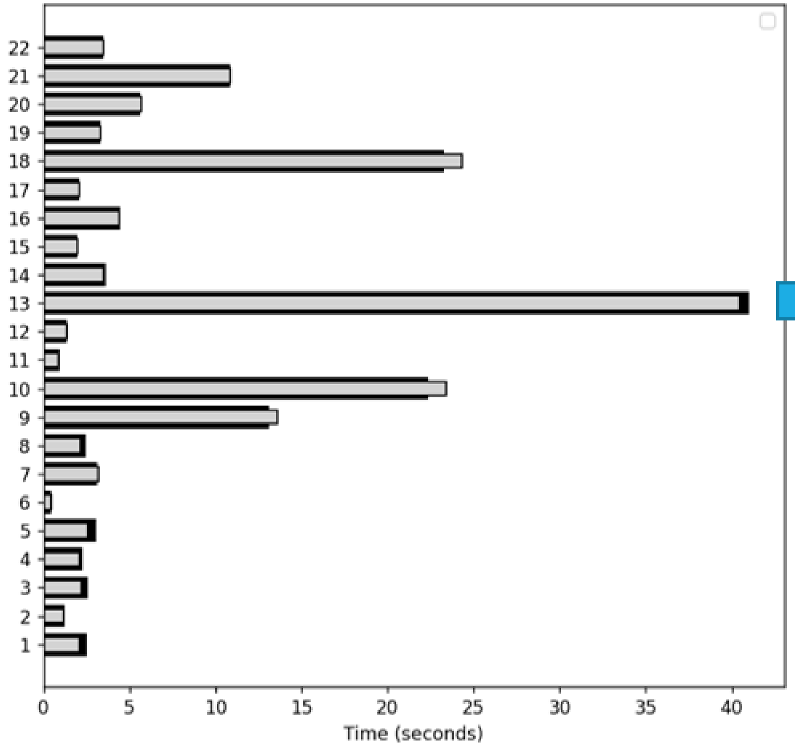


Figure 3 Query execution time for first run (black) and second run (gray) on TPC-H with SF=100

```
SELECT
    c_count,
    count(*) AS custdist
FROM (
    SELECT
        c_custkey,
        count(o_orderkey)
    FROM
        customer
    LEFT OUTER JOIN orders ON c_custkey = o_custkey
    AND o_comment NOT LIKE '%special%requests%'
GROUP BY
    c_custkey) AS c_orders (c_custkey,
        c_count)
GROUP BY
    c_count
ORDER BY
    custdist DESC,
    c_count DESC;
```

Scalability (absolute and relative)

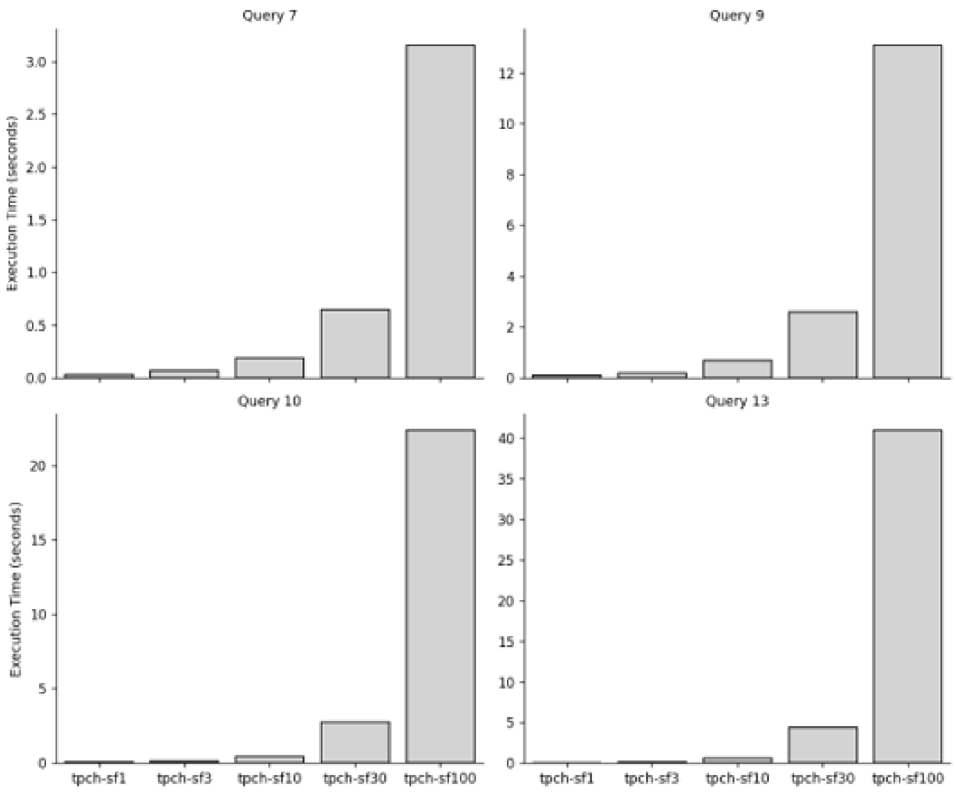


Figure 4 Query execution time for 4 selected queries on different TPC-H databases.

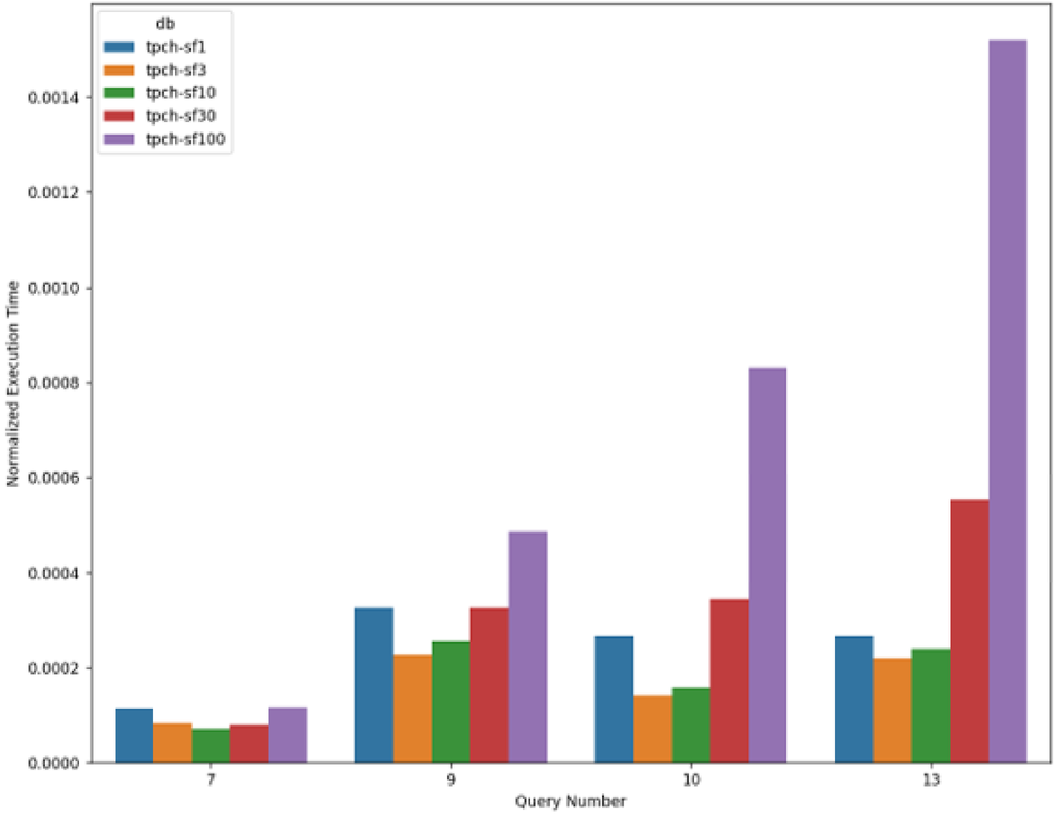


Figure 5 TPC-H query execution time divided by database size for 4 selected queries.

Comparative Analysis

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

10 groupby queries

5 join queries

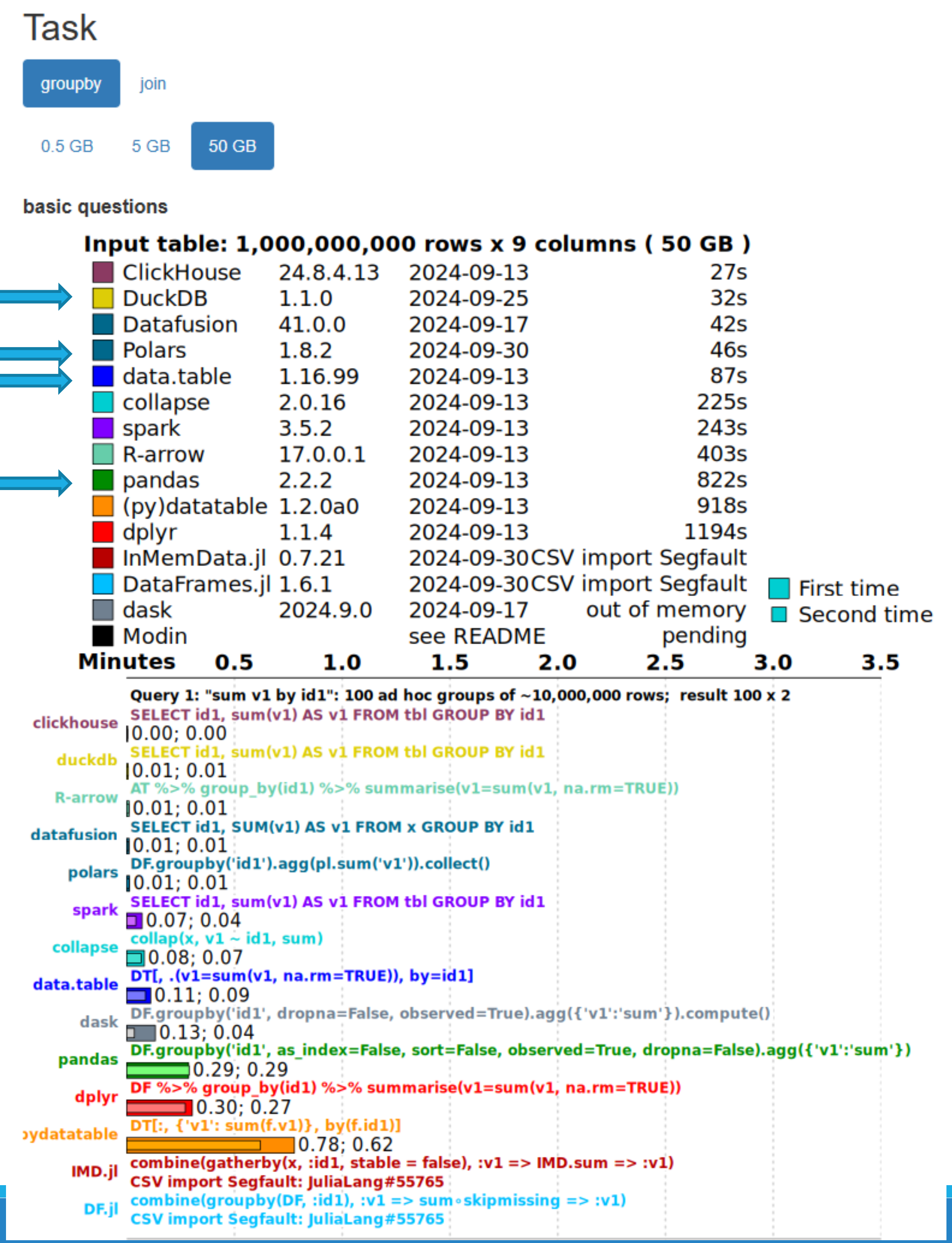
dataset size = 0,5 ; 5 ; 50 GB (.csv)

2 runs: focus on the first

runs on EC2 c6id.metal

- CPU: Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz
- CPU cores: 128
- RAM GB: 250

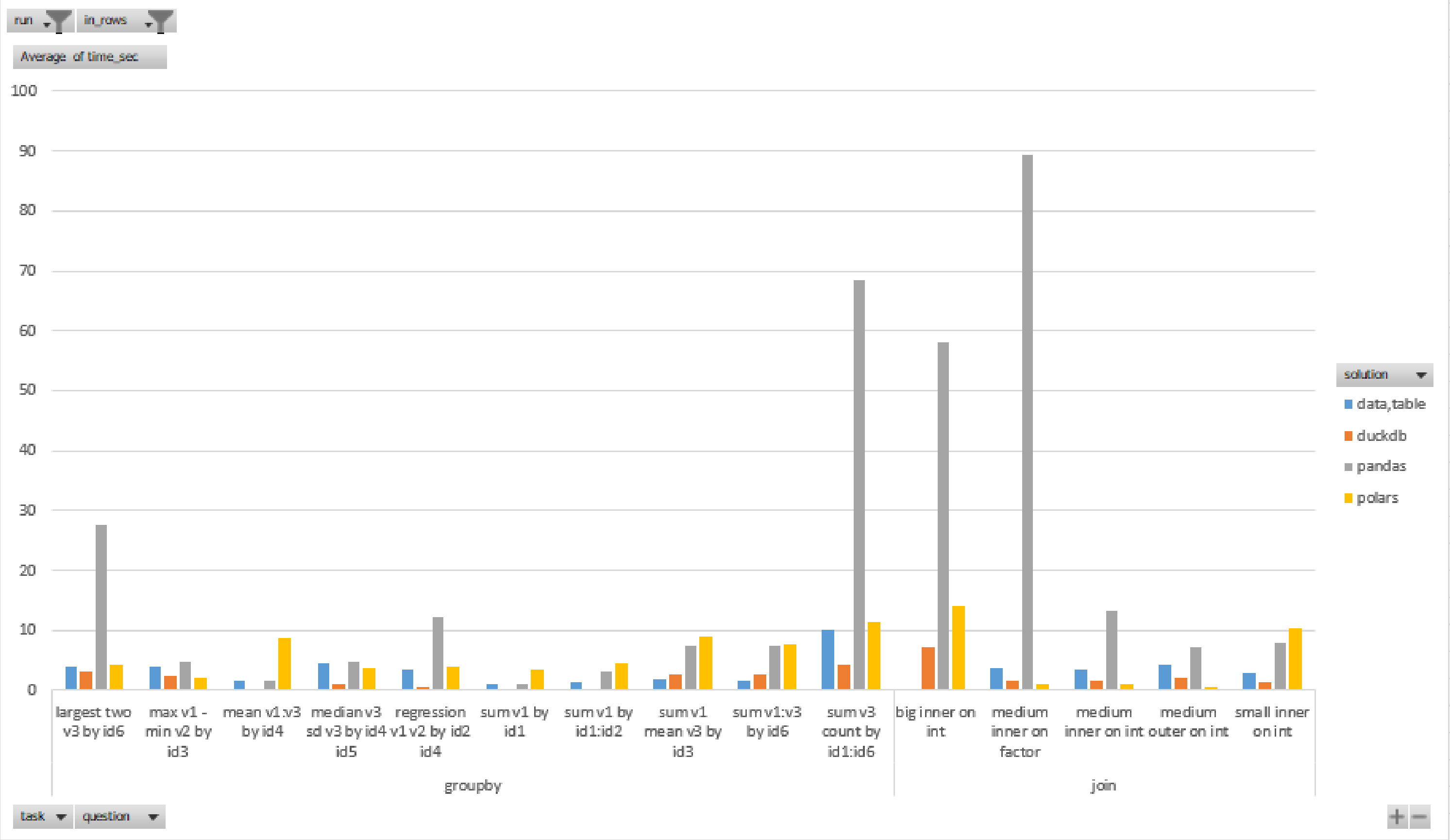
famous benchmark in DS/DA land



Conclusions

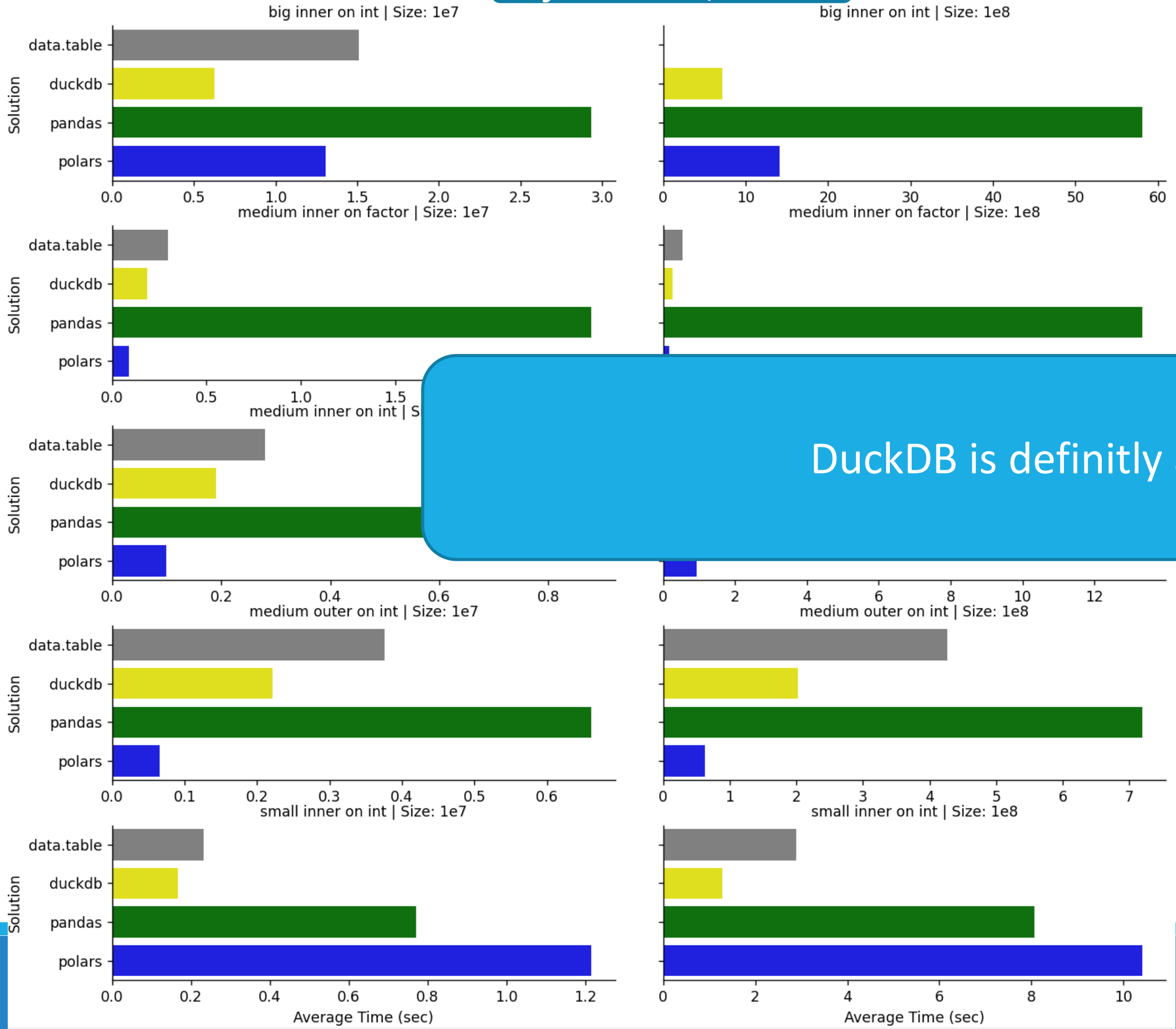
run	1
in_rows	100000000

Average of time_sec	Column Labels			
Row Labels	data,table	duckdb	pandas	polars
groupby	**3,2875**	**1,6685**	**13,8044**	**5,8665**
largest two v3 by id6	4,088	3,276	27,488	4,315
max v1 - min v2 by id3	4,036	2,295	4,802	2,018
mean v1:v3 by id4	1,65	0,088	1,483	8,644
median v3 sd v3 by id4 id5	4,362	0,946	4,881	3,618
regression v1 v2 by id2 id4	3,316	0,414	12,069	3,961
sum v1 by id1	0,952	0,044	1,039	3,439
sum v1 by id1:id2	1,177	0,188	3,218	4,498
sum v1 mean v3 by id3	1,849	2,66	7,279	9,106
sum v1:v3 by id6	1,487	2,583	7,402	7,66
sum v3 count by id1:id6	9,958	4,191	68,383	11,406
join	**3,5845**	**2,7566**	**35,2052**	**5,4499**
big inner on int		7,2065	58,076	14,1635
medium inner on factor	3,6635	1,6815	89,3655	1,1145
medium inner on int	3,519	1,582	13,326	0,941
medium outer on int	4,2625	2,02	7,1935	0,6285
small inner on int	2,893	1,293	8,065	10,402
Grand Total	**3,4195**	**2,21255**	**24,5048**	**5,6582**

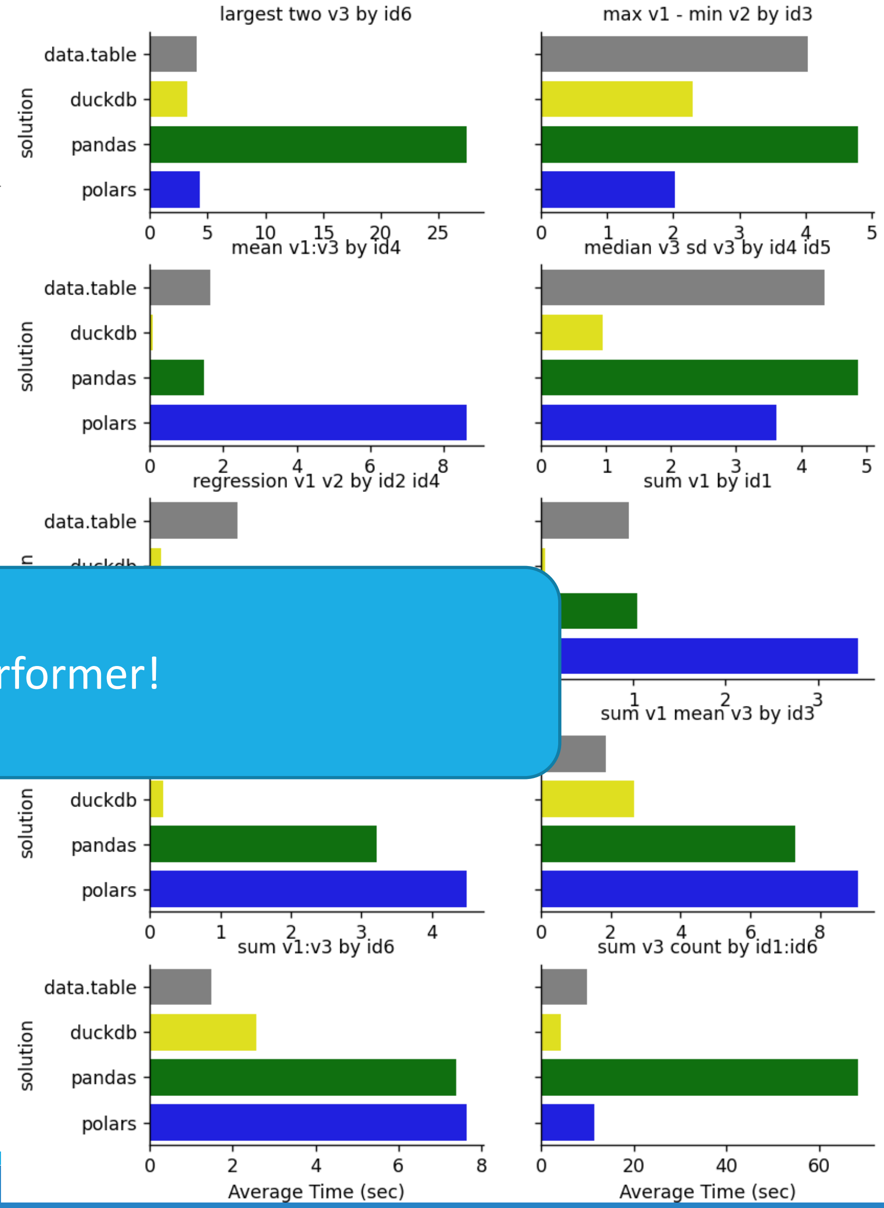


Conclusion

join 1e7 ; 1e8



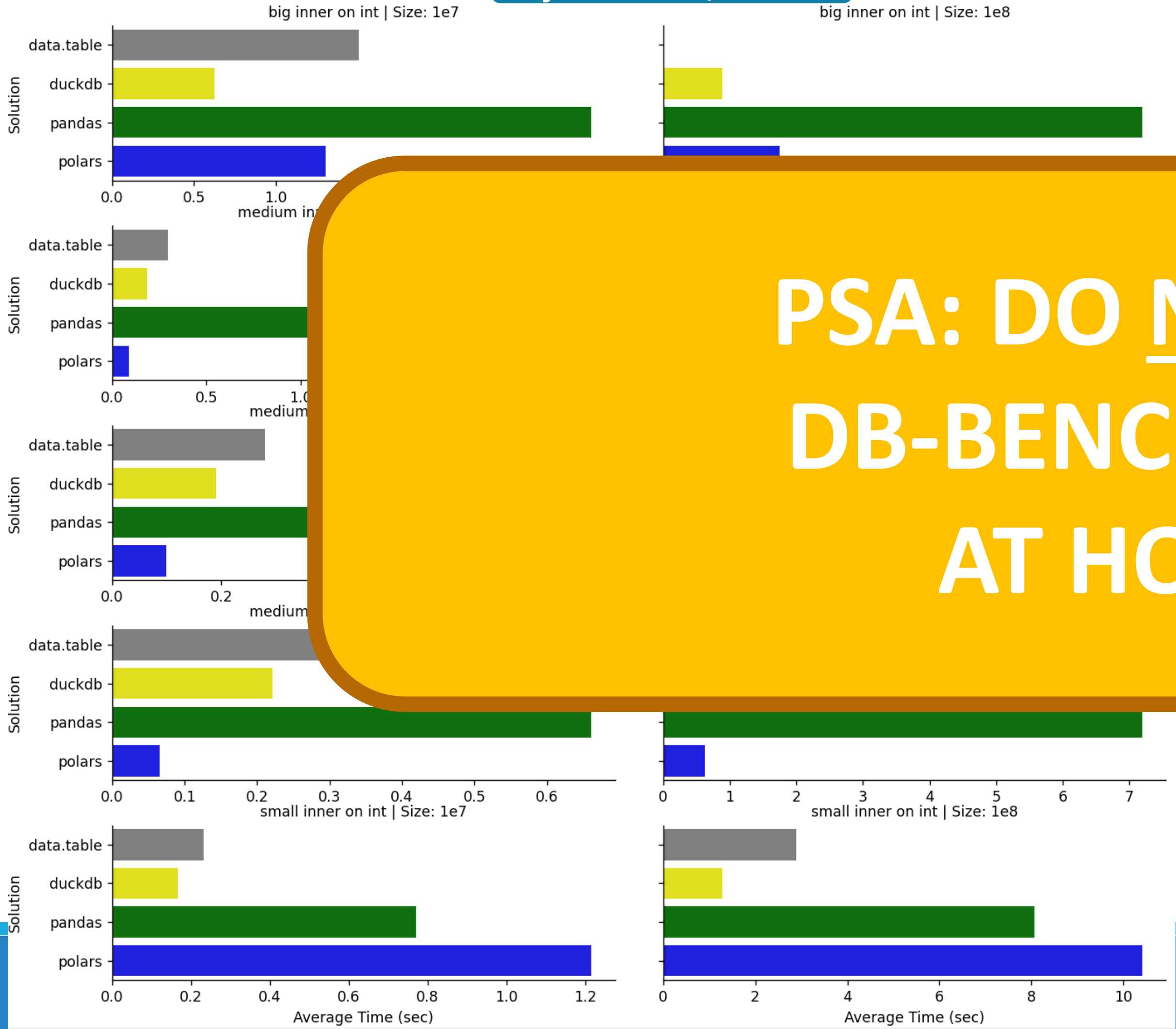
groupby 1e8



DuckDB is definitly a top performer!

Conclusion

join 1e7 ; 1e8



groupby 1e8

PSA: DO NOT TRY DB-BENCHMARK AT HOME!

Take home messages

- “Small data” is still big – hardware is impressive – try to upscale before outscaling!
- If your DBA refuses your analytical queries, he has good reason too – use an OLAP query engine!
- Analysis on 100GB datasets can easily be done on single workstations
- Ease-of-use, available bindings, API stability, documentation, ... matter too.
  But it’s always nice to be fast ;-)

Q&A