SQLFrame migrates PySpark to DuckDB without code changes

Effortlessly modernize your legacy

Nicolas Renkamp

24.06.2026 DuckCon #7

Father of two

Heidelberg-based

Leading Data Platforms at Merck
in Darmstadt

<3 DuckDB

github.com/nicornk

Nicolas Renkamp

key figures

+62,000
Employees worldwide

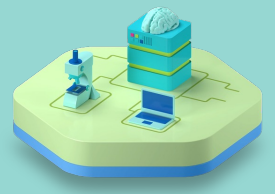
21.1
Sales (€ billion) in 2025

2.4
R&D (€ billion)

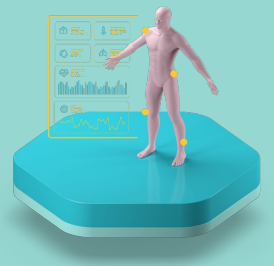
65
Countries

Founded
1668

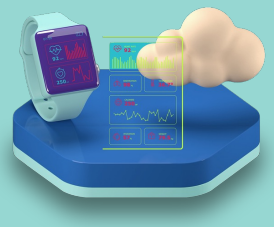
expertise



Life Science



Healthcare



Electronics

MERCK

~~Problems~~ Challenges in 2025

- ~10 years & thousands of transformations feeding data products and use cases, mainly PySpark logic

- Different level of skill sets in the engineering organisation

- FOMO

Do we need a distributed engine at all?

The Lost Decade of Small Data?

Hannes Mühleisen
2025-05-19 · 9 min

TL;DR: We benchmark DuckDB on a 2012 MacBook Pro to decide: did we lose a decade chasing distributed architectures for data analytics?

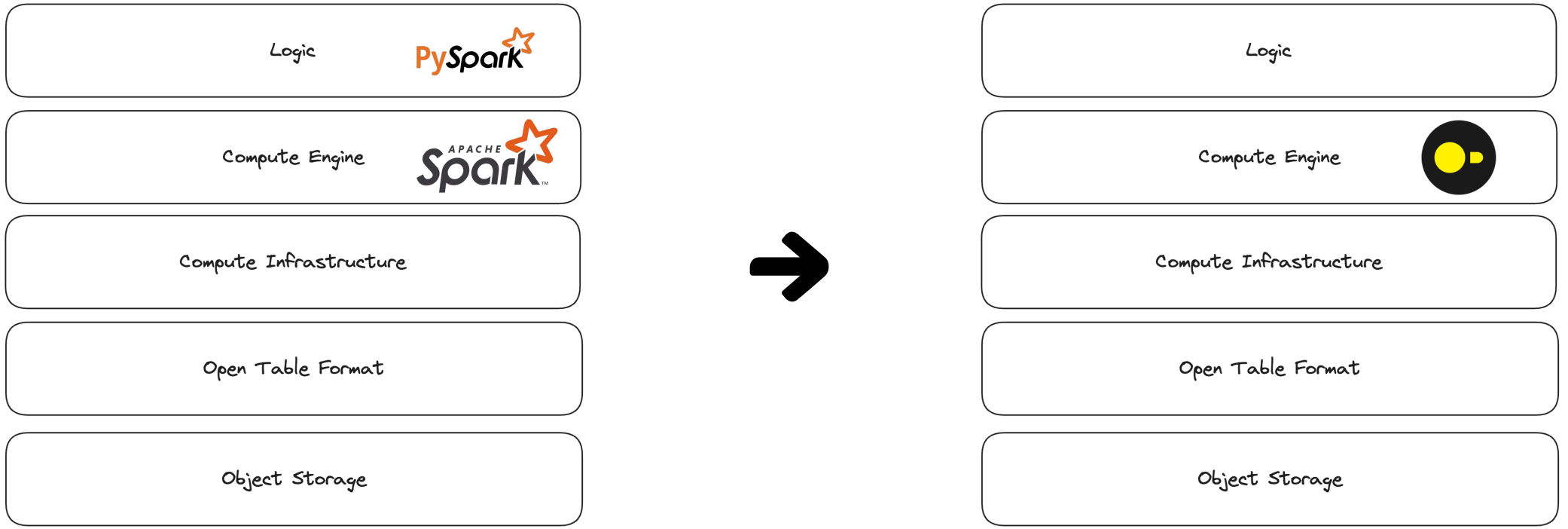
Much has been said, not in the very least by ourselves, about how data is actually not that "Big" and how the speed of hardware innovation is outpacing the growth of useful datasets. We may have gone so far to predict a data singularity in the near future, where 99% of useful datasets can be comfortably queried on a single node. As it was recently shown, the median scan in Amazon Redshift and Snowflake reads a doable 100 MB of data, and the 99.9-percentile reads less than 300 GB. So the singularity might be closer than we think.

In Merck's case, of queries that scan at least 1 MB, the **median query scans** about **55.5 MB**.

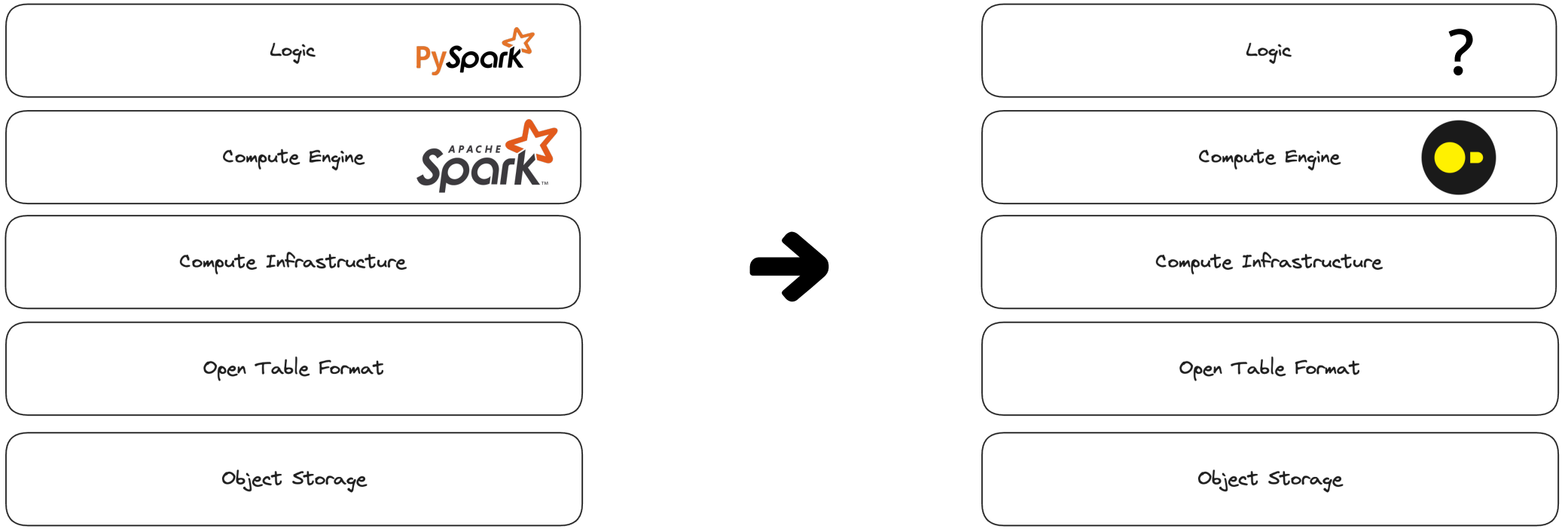
The 99.9th percentile query scans about **35.23 GB**.

Corporate Data Lake established in 2017

Running DuckDB as data transformation engine



Running DuckDB as data transformation engine



Migrating from Spark to DuckDB usually means simplifying your compute infrastructure:
From clusters to single node instances (w/ NVME SSDs)

How do I migrate all of that code?

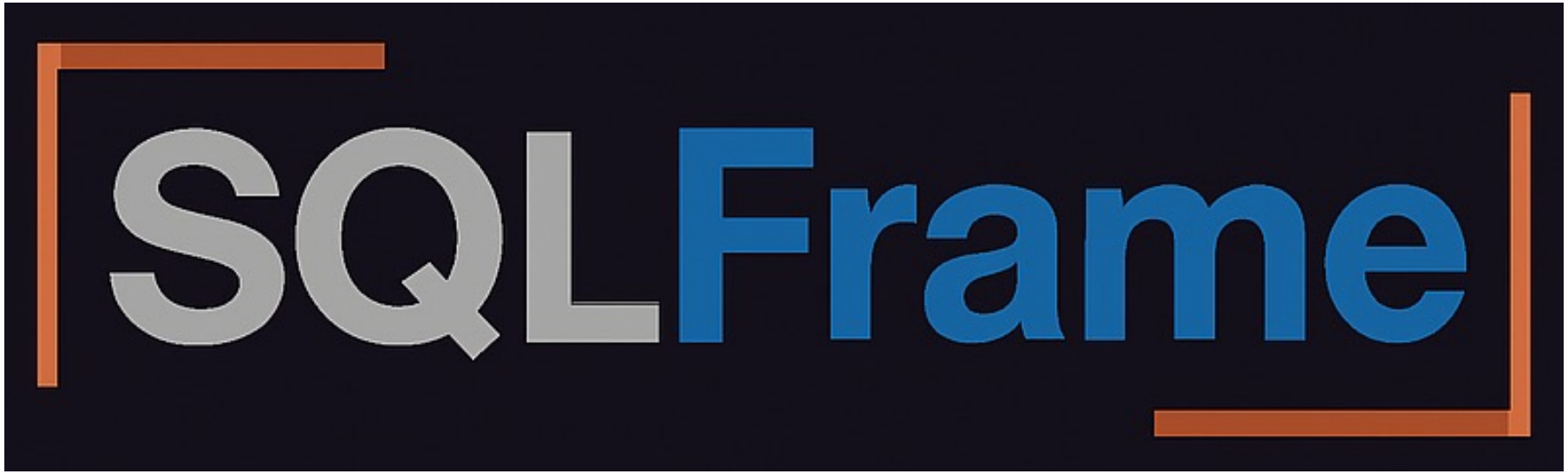
Selected Focus Areas for adoption

Highly standardized transformations

```
SELECT
    CASE WHEN TRIM(matnr) IN ('', 'null', '/', 'NaN') THEN NULL ELSE TRIM(matnr) END AS matnr,
    CASE WHEN TRIM(mtart) IN ('', 'null', '/', 'NaN') THEN NULL ELSE TRIM(mtart) END AS mtart,
    CASE WHEN TRIM(mbrsh) IN ('', 'null', '/', 'NaN') THEN NULL ELSE TRIM(mbrsh) END AS mbrsh
    -- repeat for every string column in the dataset
FROM mara;
```

Expensive jobs that shouldn't be

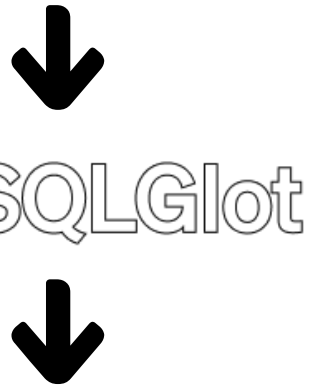
Aggregate runtime	Duration of stages	Count of stages	Count of tasks	Disk spillage	Shuffle write
70d 2h 28m 23s across all tasks	8h 0m 35s	87 + 102 skipped	33355 + 73883 skipped	5.7TB across all tasks	5.1TB



SQLFrame implements the PySpark DataFrame API in order to enable running transformation pipelines directly on database engines - no Spark clusters or dependencies required.

SQLGlot is a SQL parser, transpiler & optimizer

```sql
SELECT
    IFF(l_discount > 0.05, 'high_discount', 'normal') AS discount_category
FROM lineitem;
```



```sql
SELECT
    CASE
        WHEN l_discount > 0.05 THEN 'high_discount'
        ELSE 'normal'
    END AS discount_category
FROM lineitem;
```

SQLGlot is a SQL parser, transpiler & optimizer

```sql
SELECT
    IFF(l_discount > 0.05, 'high_discount', 'normal') AS discount_category
FROM lineitem;
```

```sql
SELECT
    CASE
        WHEN l_discount > 0.05 THEN 'high_discount'
        ELSE 'normal'
    END AS discount_category
FROM lineitem;
```

SQLMesh

Fivetran

dbt State

Greybeam

ibis-project/ibis

the portable Python dataframe library

221 Contributors · 408 Issues · 264 Discussions · 7k Stars · 730 Forks

SQLFrame adds a PySpark API to SQLGlot

```python
df = lineitem.select(
    F.when(F.col("l_discount") > 0.05, "high_discount")
    .otherwise("normal")
    .alias("discount_category")
)
```

PySpark

SQLFrame

SQLGlot

```sql
SELECT
  CASE WHEN "lineitem"."l_discount" > 0.05 THEN
'high_discount' ELSE 'normal' END AS "discount_category"
FROM "lineitem" AS "lineitem"
```

SQLFrame adds a PySpark API to SQLGlot

```python
df = lineitem.select(
    F.when(F.col("l_discount") > 0.05, "high_discount")
    .otherwise("normal")
    .alias("discount_category")
)
```

PySpark

SQLFrame

SQLGlot

```sql
SELECT
  CASE WHEN "lineitem"."l_discount" > 0.05 THEN
'high_discount' ELSE 'normal' END AS "discount_category"
FROM "lineitem" AS "lineitem"
```

Ryan Eakman

eakmanrq/sqlframe
v4.3.0 511 25

narwhals-dev/narwhals
v2.22.1 1.7k 199

Blueprint for SQLFrame + DuckDB

```python
import duckdb
from sqlframe.duckdb import DuckDBSession
from sqlframe.duckdb import functions as F
from sqlglot.dialects.dialect import NormalizationStrategy
from sqlglot.dialects.duckdb import DuckDB
from sqlglot.dialects.spark import Spark

# Ensure that the column names are consistent with the equivalent Spark code.
# Without this, column names get folded to lower case.
Spark.NORMALIZATION_STRATEGY = NormalizationStrategy.CASE_SENSITIVE
DuckDB.NORMALIZATION_STRATEGY = NormalizationStrategy.CASE_SENSITIVE

conn = duckdb.connect(database=":memory:")
spark = DuckDBSession(conn=conn)

# Spill to NVME SSD
conn.execute("SET temp_directory = '/ephemeral_data';")
# Reduce memory footprint
conn.execute("SET preserve_insertion_order = false;")

conn.execute("""CREATE OR REPLACE SECRET secret (
                    TYPE s3,
                    REGION 'eu-central-1',
                    PROVIDER credential_chain);
""")
conn.execute("""
    CREATE VIEW lineitem AS
        SELECT * FROM READ_PARQUET('s3://<bucket-name>/lineitem/*.parquet');
""")

lineitem = spark.table("lineitem")

# User Code
df = lineitem.select(
    F.when(F.col("l_discount") > 0.05, "high_discount")
    .otherwise("normal")
    .alias("discount_category")
)
# User Code - End

sql = df.sql(dialect="duckdb")

conn.execute(
    f"COPY ({sql}) TO 's3://<bucket-name>/results/output.parquet' (OVERWRITE true);"
)
```

Platform Engineering Team establishes connectivity, configures defaults and sets guardrails

Data Engineering Team writes data transformation logic

Blueprint for SQLFrame + DuckDB

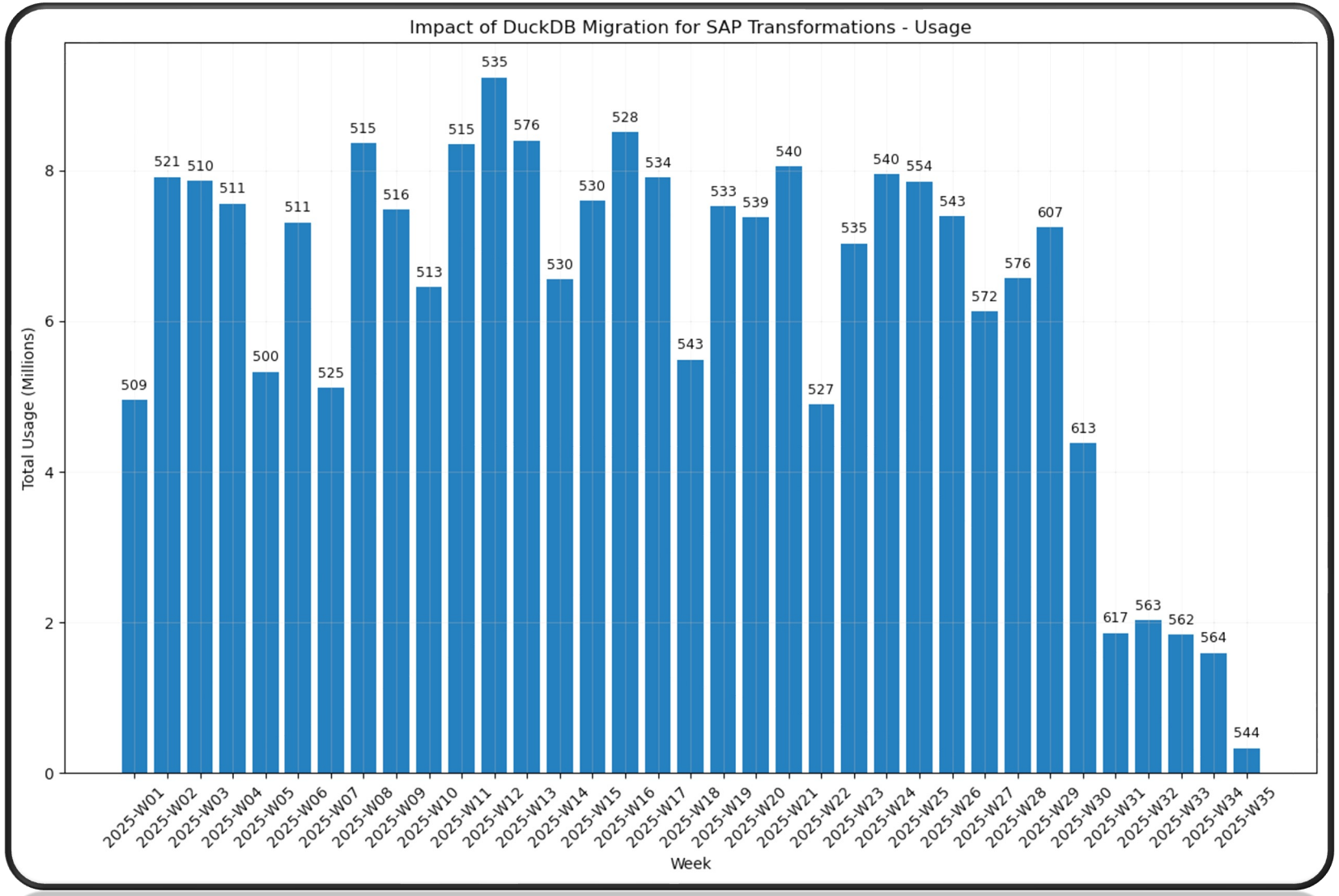
```python
import duckdb
from sqlframe.duckdb import DuckDBSession
from sqlframe.duckdb import functions as F
from sqlglot.dialects.dialect import NormalizationStrategy
from sqlglot.dialects.duckdb import DuckDB
from sqlglot.dialects.spark import Spark

# Ensure that the column names are consistent with the equivalent Spark code.
# Without this, column names get folded to lower case.
Spark.NORMALIZATION_STRATEGY = NormalizationStrategy.CASE_SENSITIVE
DuckDB.NORMALIZATION_STRATEGY = NormalizationStrategy.CASE_SENSITIVE

conn = duckdb.connect(database=":memory:")
spark = DuckDBSession(conn=conn)

# Spill to NVME SSD
conn.execute("SET temp_directory = '/ephemeral_data';")
# Reduce memory footprint
conn.execute("SET preserve_insertion_order = false;")

conn.execute("""CREATE OR REPLACE SECRET secret (
                    TYPE s3,
                    REGION 'eu-central-1',
                    PROVIDER credential_chain);
""")
conn.execute("""
    CREATE VIEW lineitem AS
        SELECT * FROM READ_PARQUET('s3://<bucket-name>/lineitem/*.parquet');
""")

lineitem = spark.table("lineitem")

# User Code
df = lineitem.select(
    F.when(F.col("l_discount") > 0.05, "high_discount")
    .otherwise("normal")
    .alias("discount_category")
)
# User Code - End

sql = df.sql(dialect="duckdb")

conn.execute(
    f"COPY ({sql}) TO 's3://<bucket-name>/results/output.parquet' (OVERWRITE true);"
)
```

Platform Engineering Team establishes connectivity, configures defaults and sets guardrails

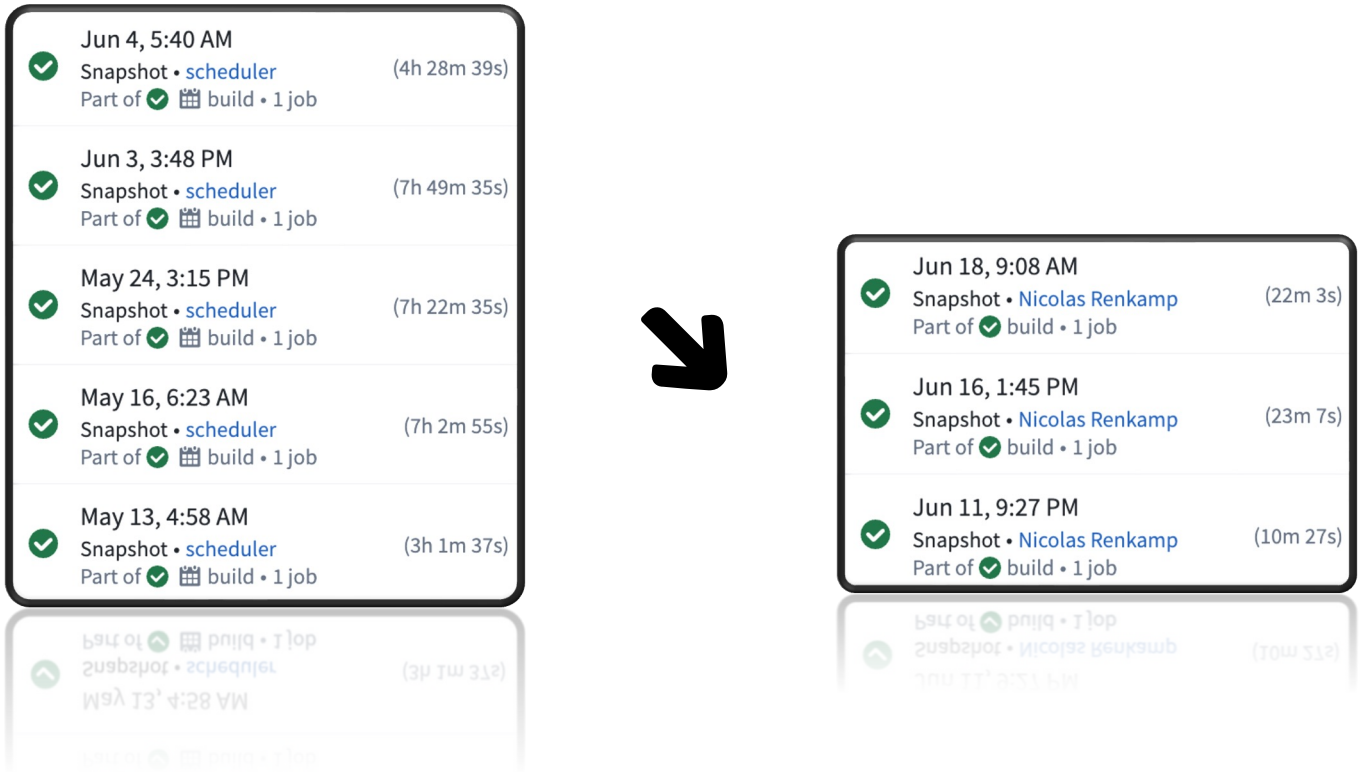
Data Engineering Team writes data transformation logic

Results

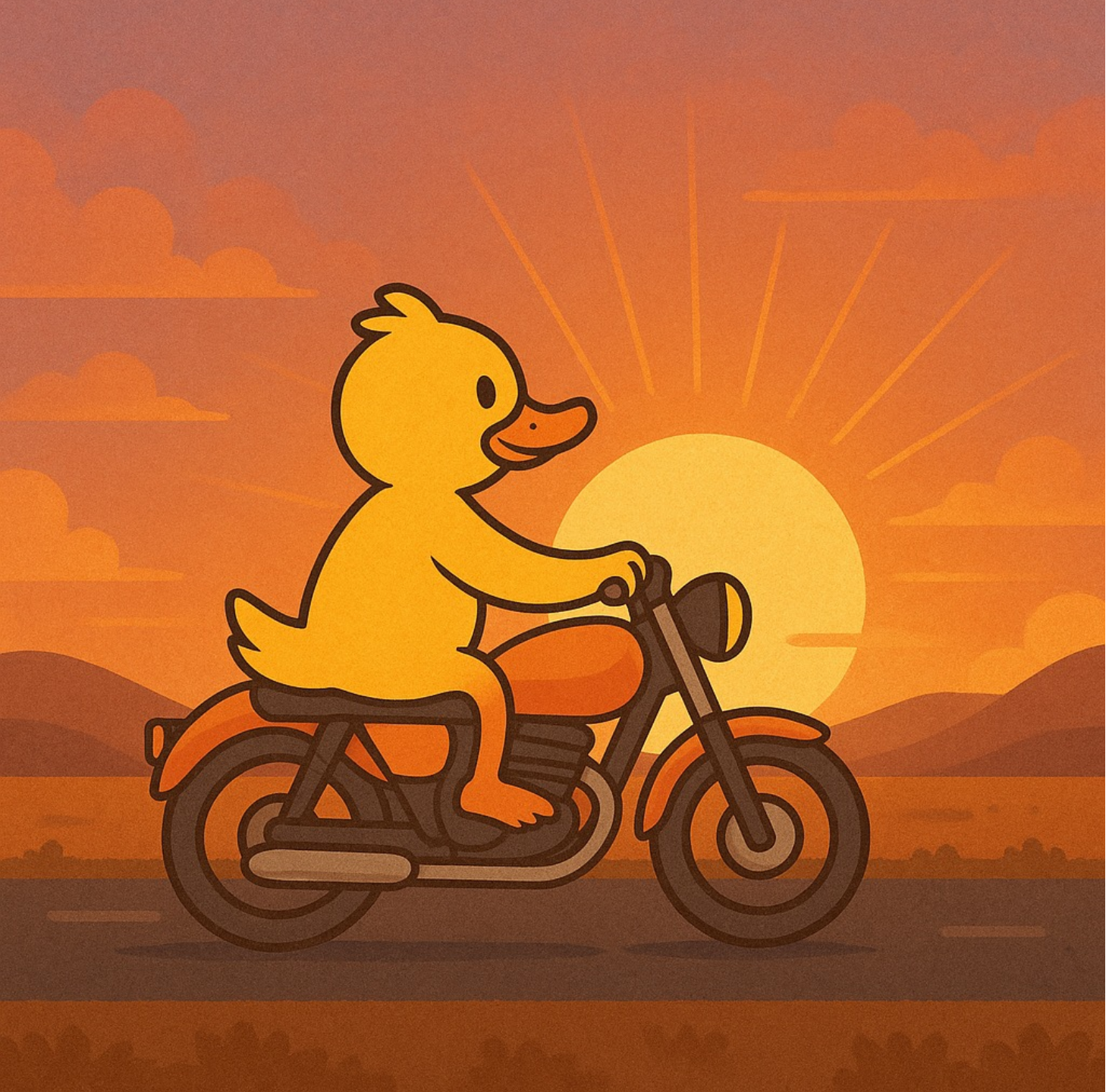
Production workload impact examples (1)



Production workload impact examples (2)



18x UNION ALLs, 5 LEFT JOINs, in total 11.5k lines of SQL Query



Why it worked for us

✅ SQLFrame to go quickly from PySpark to DuckDB SQL

✅ Focus on specific workloads with most promising performance improvements

✅ Keep existing architecture & governance approaches unchanged

✅ DuckDB as start into a more sovereign data stack, extend into other layers from there (e.g. DuckLake)

Take the DuckDB pill…

Nicolas Renkamp
linkedin.com/in/nicolasrenkamp
@nicornk

Backup

It‘s 2026: Why not use LLMs to convert the logic to SQL?