zoology 101: python, pandas and duckdb

presenter:
Jaroslav Bezdek

date:
16 November 2024

01 motivation

MY_TOOLBOX

MY_TOOLBOX

MY_TOOLBOX

MY_TOOLBOX


pandas


scikit
learn


matplotlib


NumPy


TensorFlow


dbt

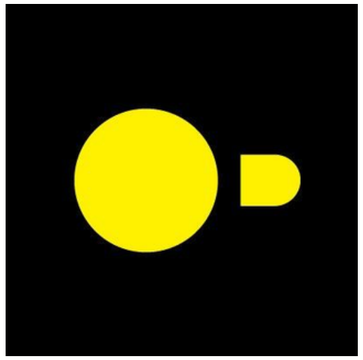

OpenAI


Streamlit

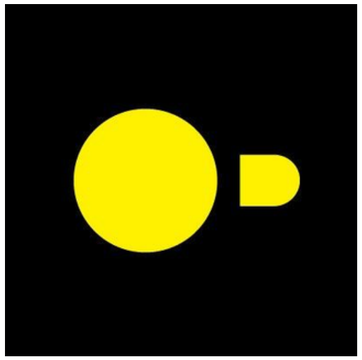
DUCKDB_INTEREST_OVER_TIME

Source: Google Trends

MY_TOOLBOX_WITH_DUCKDB



PANDAS_VS_DUCKDB



02 data

```
>> print(pinguins)

shape: (344, 6)
```

pinguin_id	date	species	bill_length	bill_depth	sex
str	str	str	f64	f64	str
N1A1	2007-11-11	adelie	39.1	18.7	MALE
N1A2	2007-11-11	adelie	39.5	17.4	FEMALE
N2A1	2007-11-16	adelie	40.3	18.0	FEMALE
N2A2	2007-11-16	adelie	null	null	null
N3A1	2007-11-16	adelie	36.7	19.3	FEMALE
…	…	…	…	…	…
N38A2	2009-12-01	gentoo	null	null	null
N39A1	2009-11-22	gentoo	46.8	14.3	FEMALE
N39A2	2009-11-22	gentoo	50.4	15.7	MALE
N43A1	2009-11-22	gentoo	45.2	14.8	FEMALE
N43A2	2009-11-22	gentoo	49.9	16.1	MALE

03 import pandas

```python
df = pd.read_csv(
    filepath_or_buffer=DATA_PATH,
)

print(df)

#     pinguin_id        date species  bill_length  bill_depth     sex
# 0         N1A1  2007-11-11  adelie         39.1        18.7    MALE
# 1         N1A2  2007-11-11  adelie         39.5        17.4  FEMALE
# 2         N2A1  2007-11-16  adelie         40.3        18.0  FEMALE
# 3         N2A2  2007-11-16  adelie          NaN         NaN     NaN
# 4         N3A1  2007-11-16  adelie         36.7        19.3  FEMALE
# ..         ...         ...     ...          ...         ...     ...
# 339      N38A2  2009-12-01  gentoo          NaN         NaN     NaN
# 340      N39A1  2009-11-22  gentoo         46.8        14.3  FEMALE
# 341      N39A2  2009-11-22  gentoo         50.4        15.7    MALE
# 342      N43A1  2009-11-22  gentoo         45.2        14.8  FEMALE
# 343      N43A2  2009-11-22  gentoo         49.9        16.1    MALE
#
# [344 rows x 6 columns]
```

```python
print(
    df
    .assign(
        sex=lambda _df: _df.sex.str.lower(),
    )
    .groupby(by="species")
    .agg(
        samples_cnt=("pinguin_id", "count"),
        male_cnt=("sex", lambda x: x[x == "male"].count()),
        bill_length_avg=("bill_length", "mean"),
    )
    .round(1)
)

#            samples_cnt  male_cnt  bill_length_avg
# species
# adelie             152        73             38.8
# chinstrap           68        34             48.8
# gentoo             124        61             47.5
```

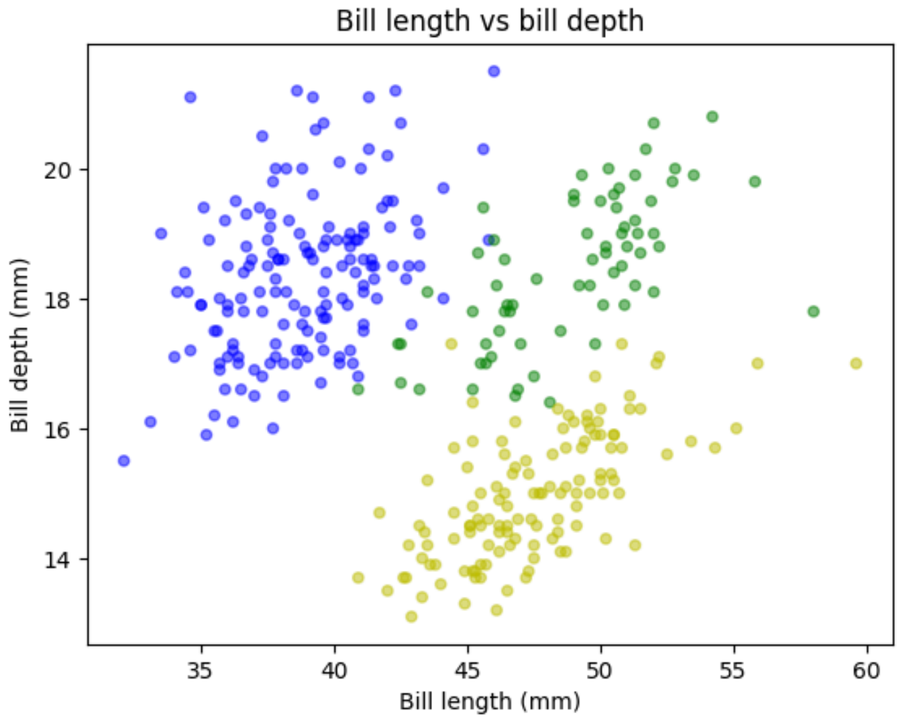
```python
# convert string to datetime
df["date"] = pd.to_datetime(df.date)

# get unique sample dates from year 2007
unique_dates_2007 = (
    df
    .date
    .loc[df.date.dt.year == 2007]
    .drop_duplicates()
    .sort_values()
    .reset_index(drop=True)
)

# get the difference between sample days
print(unique_dates_2007 - unique_dates_2007.shift())

# 0       NaT
# 1    1 days
# 2    1 days
# ..
# 14   1 days
# 15   1 days
# 16   3 days
```

```python
(
    df
    .plot(
        kind="scatter",
        x="bill_length",
        y="bill_depth",
        color=df.species.map({
            "adelie": "b",
            "chinstrap": "g",
            "gentoo": "y",
        }),
        alpha=0.5,
        xlabel="Bill length (mm)",
        ylabel="Bill depth (mm)",
        title="Bill length vs bill depth",
    )
)
```



STRENGTHS_AND_WEAKNESSES

- **Swiss army knife of data analytics**

STRENGTHS_AND_WEAKNESSES

- Swiss army knife of data analytics
- Integration with other Python libraries

STRENGTHS_AND_WEAKNESSES

- Swiss army knife of data analytics
- Integration with other Python libraries
- Community and documentation

STRENGTHS_AND_WEAKNESSES

- Swiss army knife of data analytics
- Integration with other Python libraries
- Community and documentation

- Performance with large datasets

STRENGTHS_AND_WEAKNESSES

- Swiss army knife of data analytics
- Integration with other Python libraries
- Community and documentation

- Performance with large datasets
- Focus on structured data

STRENGTHS_AND_WEAKNESSES

- Swiss army knife of data analytics
- Integration with other Python libraries
- Community and documentation

- Performance with large datasets
- Focus on structured data
- Steeper learning curve for advanced features

04 import duckdb

“DuckDB is a fast in-process analytical database.”

—

duckdb.org

DATABASE_WORKLOAD_TYPES

workload type	

Source: Kojo Osei

DATABASE_WORKLOAD_TYPES

workload type
transactional

Source: Kojo Osei

DATABASE_WORKLOAD_TYPES

workload type	example query
transactional	update users table with new user object after sigup

Source: Kojo Osei

DATABASE_WORKLOAD_TYPES

workload type	example query	example database
transactional	update users table with new user object after sigup	Postgres, SQLite

Source: Kojo Osei

DATABASE_WORKLOAD_TYPES

workload type	example query	example database
transactional	update users table with new user object after sigup	Postgres, SQLite
analytical		

Source: Kojo Osei

DATABASE_WORKLOAD_TYPES

workload type	example query	example database
transactional	update users table with new user object after sigup	Postgres, SQLite
analytical	how many users have signed up in the last 2 weeks, broken by age and location?	

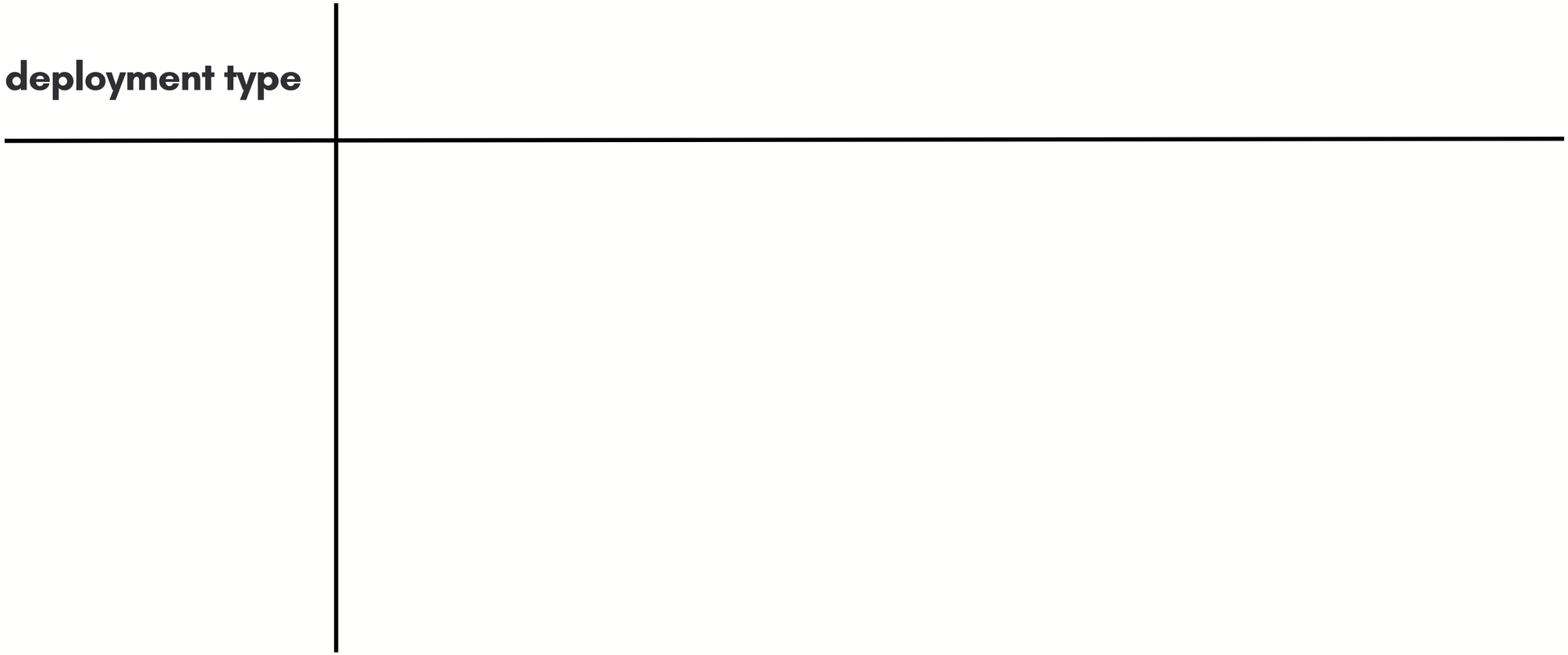
Source: Kojo Osei

DATABASE_WORKLOAD_TYPES

workload type	example query	example database
transactional	update users table with new user object after sigup	Postgres, SQLite
analytical	how many users have signed up in the last 2 weeks, broken by age and location?	Snowflake, BigQuery

Source: Kojo Osei

DATABASE_DEPLOYMENT_TYPES



Source: Kojo Osei

DATABASE_DEPLOYMENT_TYPES

deployment type
stand-alone

Source: Kojo Osei

DATABASE_DEPLOYMENT_TYPES

deployment type	example database
stand-alone	Snowflake, Postgres, BigQuery

Source: Kojo Osei

DATABASE_DEPLOYMENT_TYPES

deployment type	example database
stand-alone	Snowflake, Postgres, BigQuery
embedded	

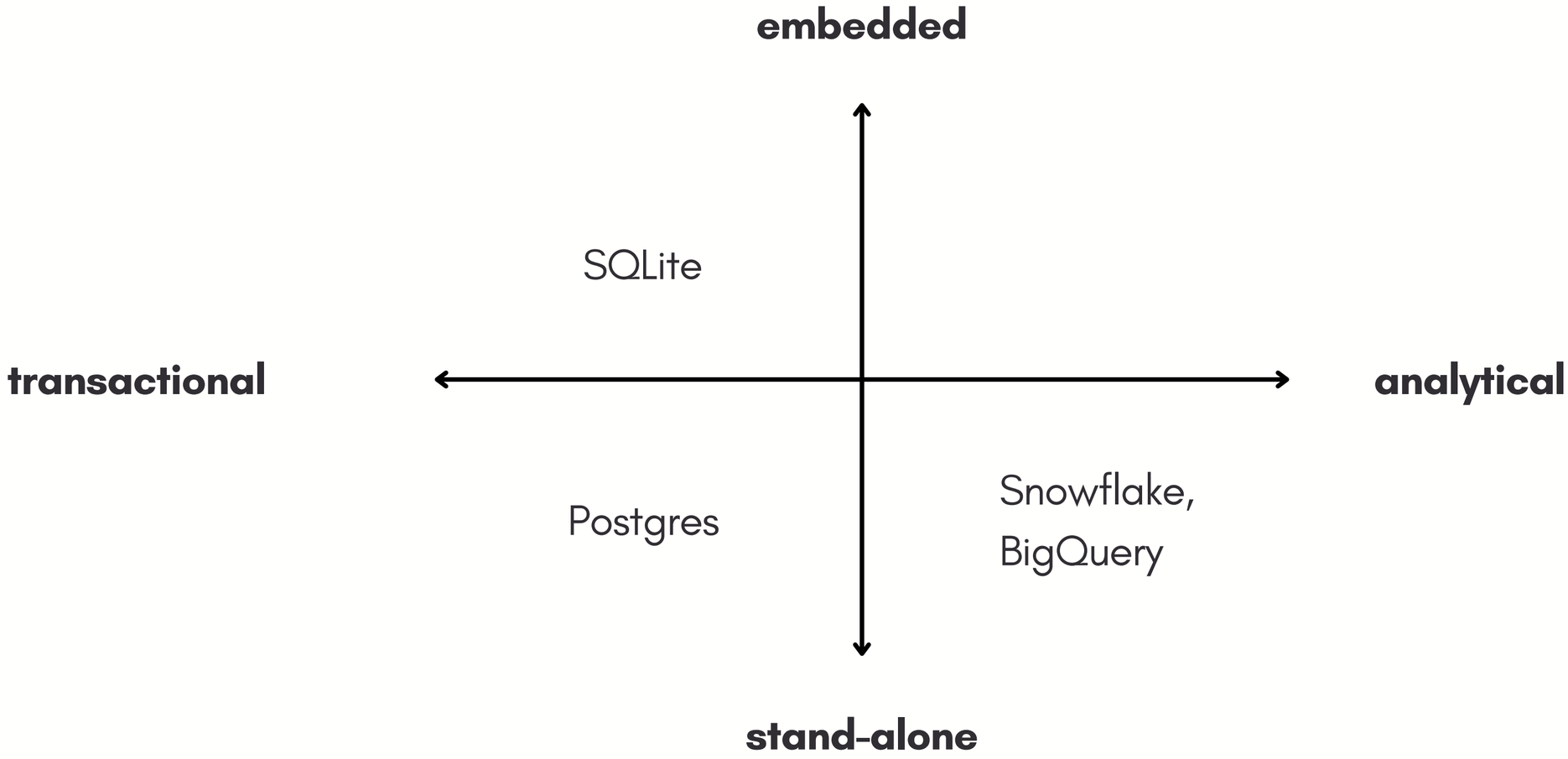
Source: Kojo Osei

DATABASE_DEPLOYMENT_TYPES

deployment type	example database
stand-alone	Snowflake, Postgres, BigQuery
embedded	SQLite

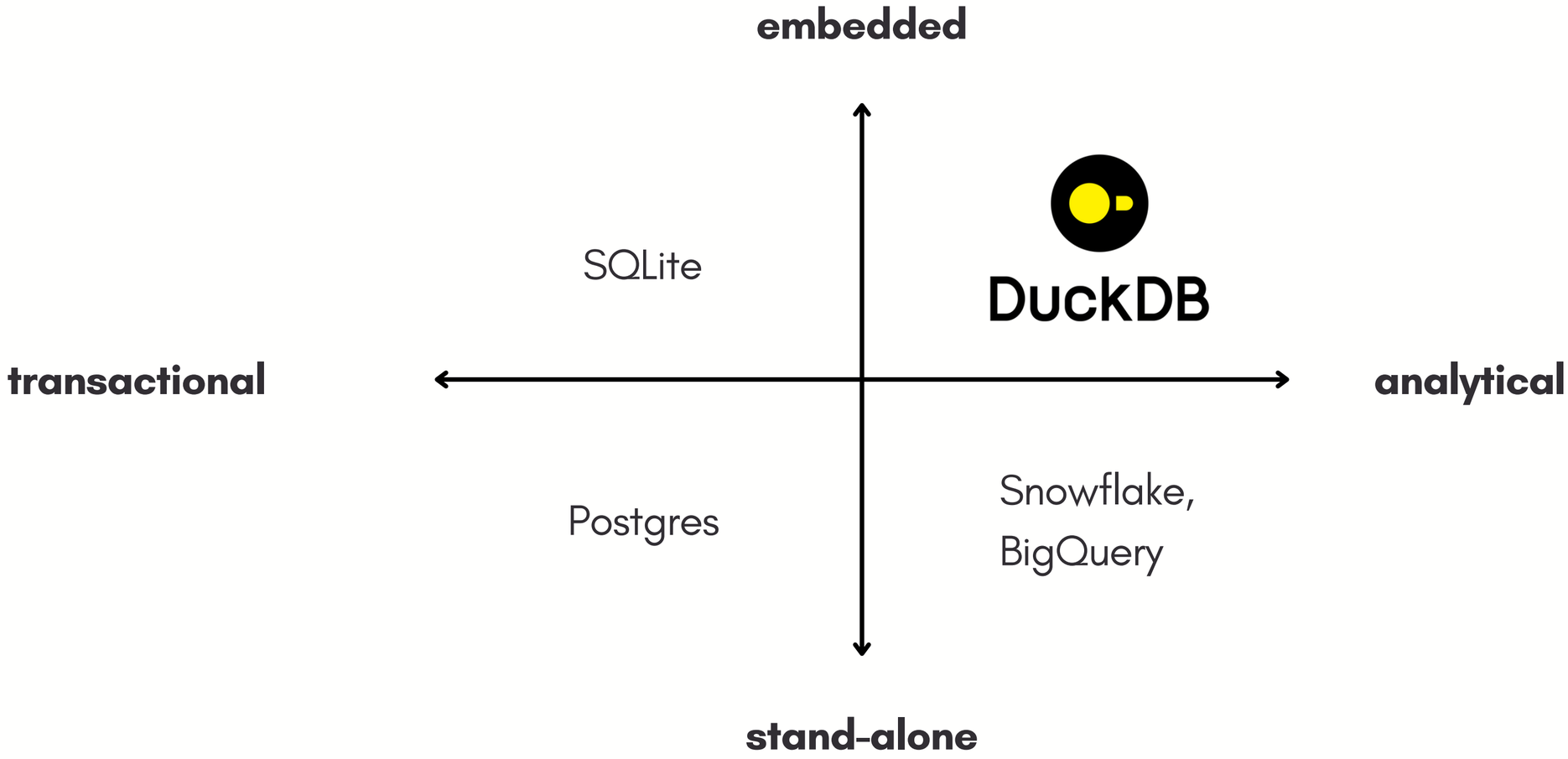
Source: Kojo Osei

DATABASE_TYPES



Source: Kojo Osei

DATABASE_TYPES



Source: Kojo Osei

```python
import duckdb

data_raw = (
    duckdb
    .read_csv(name=DATA_PATH)
    .select("species", "bill_length", "bill_depth", "sex")
)

print(data_raw)

# ┌─────────┬──────────────┬─────────────┬─────────┐
# │ species │ bill_length  │ bill_depth  │   sex   │
# │ varchar │    double    │   double    │ varchar │
# ├─────────┼──────────────┼─────────────┼─────────┤
# │ adelie  │         39.1 │        18.7 │ MALE    │
# │ adelie  │         39.5 │        17.4 │ FEMALE  │
# │   .     │           .  │          .  │  .      │
# │ gentoo  │         45.2 │        14.8 │ FEMALE  │
# │ gentoo  │         49.9 │        16.1 │ MALE    │
# ├─────────┴──────────────┴─────────────┴─────────┤
# │ 344 rows (20 shown)                4 columns  │
# └───────────────────────────────────────────────┘
```

```
(
    data_raw
    .project("""
        species,
        lower(sex) as sex,
        bill_length,
        bill_depth
    """)
    .filter("sex = 'male'")
    .aggregate("""
        species,
        avg(bill_length)::int as bill_length_avg,
        avg(bill_depth)::int as bill_depth_avg
    """)
    .order("species")
)

# ┌─────────────┬─────────────────┬─────────────────┐
# │   species   │ bill_length_avg │ bill_depth_avg  │
# │   varchar   │      int32      │      int32      │
# ├─────────────┼─────────────────┼─────────────────┤
# │ adelie      │              40 │              19 │
# │ chinstrap   │              51 │              19 │
# │ gentoo      │              49 │              16 │
# └─────────────┴─────────────────┴─────────────────┘
```

```
(
    data_raw
    .project("""
        species,
        lower(sex) as sex,
        bill_length,
        bill_depth
    """)
    .filter("sex = 'male'")
    .aggregate("""
        species,
        avg(bill_length)::int as bill_length_avg,
        avg(bill_depth)::int as bill_depth_avg
    """)
    .order("species")
)
```

```
# ┌─────────────┬─────────────────┬─────────────────┐
# │   species   │ bill_length_avg │ bill_depth_avg  │
# │   varchar   │      int32      │      int32      │
# ├─────────────┼─────────────────┼─────────────────┤
# │ adelie      │              40 │              19 │
# │ chinstrap   │              51 │              19 │
# │ gentoo      │              49 │              16 │
# └─────────────┴─────────────────┴─────────────────┘
```

```
duckdb.sql(query=f"""
    with cleaned as (
        select
            species,
            lower(sex) as sex,
            bill_length,
            bill_depth
        from data_raw
        where sex is not null
    ),

    agg as (
        select
            species,
            avg(bill_length)::int as bill_length_avg,
            avg(bill_depth)::int as bill_depth_avg
        from cleaned
        where sex = 'male'
        group by species
        order by species
    )

    select * from agg;
    """
)
```

```
# ┌─────────────┬─────────────────┬─────────────────┐
# │   species   │ bill_length_avg │ bill_depth_avg  │
# │   varchar   │      int32      │      int32      │
# ├─────────────┼─────────────────┼─────────────────┤
# │ adelie      │              40 │              19 │
# │ chinstrap   │              51 │              19 │
# │ gentoo      │              49 │              16 │
# └─────────────┴─────────────────┴─────────────────┘
```

```python
df_agg = agg.to_df()
print(df_agg)

#       species  bill_length_avg  bill_depth_avg
# 0      adelie               40              19
# 1   chinstrap               51              19
# 2      gentoo               49              16

agg = duckdb.sql("select * from df_agg")
print(agg)

# ┌─────────────┬─────────────────┬────────────────┐
# │   species   │ bill_length_avg │ bill_depth_avg │
# │   varchar   │      int32      │     int32      │
# ├─────────────┼─────────────────┼────────────────┤
# │ adelie      │              40 │             19 │
# │ chinstrap   │              51 │             19 │
# │ gentoo      │              49 │             16 │
# └─────────────┴─────────────────┴────────────────┘
```

STRENGTHS_AND_WEAKNESSES

- **Performance**

STRENGTHS_AND_WEAKNESSES

- **Performance**
- **Reading multiple files at once**

STRENGTHS_AND_WEAKNESSES

- Performance
- Reading multiple files at once
- SQL syntax

STRENGTHS_AND_WEAKNESSES

- Performance
- Reading multiple files at once
- SQL syntax
- APIs for major programming languages

STRENGTHS_AND_WEAKNESSES

👍

- Performance
- Reading multiple files at once
- SQL syntax
- APIs for major programming languages

👎

- “Only” analytical workloads

STRENGTHS_AND_WEAKNESSES

- Performance
- Reading multiple files at once
- SQL syntax
- APIs for major programming languages

- “Only” analytical workloads
- Community support

STRENGTHS_AND_WEAKNESSES

- Performance
- Reading multiple files at once
- SQL syntax
- APIs for major programming languages

- “Only” analytical workloads
- Community support
- Immaturity

05 comparison

COMPARISON

	🐼	🦆

COMPARISON

		
speed	slow	fast

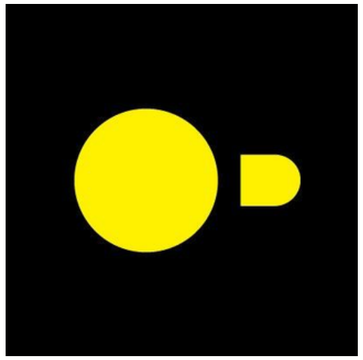
COMPARISON

		
speed	slow	fast
size	big	small

COMPARISON

		
speed	slow	fast
size	big	small
maturity	adult	teenager

MY_TOOLBOX_ENRICHED_BY_DUCKDB



Thank You