Pandas on DuckDB
With Ponder

Aditya Parameswaran
Cofounder and President, Ponder
Associate Professor, UC Berkeley
The Swiss Army Knife of Data Science/ML
Python explosion blamed on pandas
Data science fad just won't die

Not content to bait developers by declaring that Python is the fastest-growing major programming language, coding community site Stack Overflow has revealed the reason for its metastasis.
The Swiss Army Knife of Data Science/ML
The Swiss Army Knife of Data Science/ML

Pandas is now as popular as Python was in 2016

Meet the man behind the most important tool in data science
Why? A Flexible & Convenient Dataframe Data Model

<table>
<thead>
<tr>
<th>Continent</th>
<th>GDP</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tabular Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>object</th>
<th>int</th>
<th>int</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continent</td>
<td>GDP</td>
<td>Population</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabular Data
Why? A Flexible & Convenient Dataframe Data Model

<table>
<thead>
<tr>
<th>object</th>
<th>int</th>
<th>int</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continent</td>
<td>GDP</td>
<td>Population</td>
</tr>
</tbody>
</table>

Tabular Data

Ordered Rows

Ordered Columns
Why? A Flexible & Convenient Dataframe Data Model

<table>
<thead>
<tr>
<th></th>
<th>Continent</th>
<th>GDP</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>China</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Germany</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabular Data

Row Labels

- USA
- China
- Japan
- Germany

Columns: Ordered
- Continent
- GDP
- Population

Rows: Ordered
### Why? A Flexible & Convenient Dataframe Data Model

- **Row Labels**: USA, China, Japan, Germany
- **Columns**: Continent, GDP, Population
- **Data Types**:
  - USA: Continent: object, GDP: int, Population: int
  - China: Continent: object, GDP: int, Population: int
  - Japan: Continent: object, GDP: int, Population: int
  - Germany: Continent: object, GDP: int, Population: int

- **Supports mixed types / column**
- **Schema not required upfront**
- **Ordered Rows**
- **Ordered Columns**

+ Hierarchical row/column labels (not shown)
Why? (II) An Expressive & Concise Data Science API

600+ functions to clean, featurize, explore, and summarize data spanning rel., linear, & spreadsheet algebra
Why? (II) An Expressive & Concise Data Science API

600+ functions to clean, featurize, explore, and summarize data spanning rel., linear, & spreadsheet algebra

With operations such as:
• Drop columns with null values
• Transpose
• One hot encoding
• Positional updates
• Elementwise matrix ops
• …
But Pandas Doesn’t Scale!

знаком Single threaded
But Pandas Doesn’t Scale!

⚠️ Single threaded

-unused CPUs

Dataset size restricted to memory of a machine
But Pandas Doesn’t Scale!

- Single threaded
- Dataset size restricted to memory of a machine
- Inefficient with memory
But Pandas Doesn’t Scale!

- Single threaded
- Dataset size restricted to memory of a machine
- Inefficient with memory
- No real optimization
But Pandas Doesn’t Scale!

- Single threaded
- Dataset size restricted to memory of a machine
- Inefficient with memory
- No real optimization

Memory of One Machine

Unused CPUs
A “Drop-in” Scalable Version of Pandas

```python
# import pandas as pd
import modin.pandas as pd
```
A “Drop-in” Scalable Version of Pandas

```python
# import pandas as pd
import modin.pandas as pd
```

https://github.com/modin-project/modin
A “Drop-in” Scalable Version of Pandas

import pandas as pd
import modin.pandas as pd

https://github.com/modin-project/modin

# pandas API

MODIN

Ray
Dask
...

10+ Million Downloads (1M+/month)
100+ contributors
Used by 40+ companies & orgs
Part of AWS, Intel distros

#100+ contributors
# Used by 40+ companies & orgs
# Part of AWS, Intel distros

https://github.com/modin-project/modin
A “Drop-in” Scalable Version of Pandas

```python
# import pandas as pd
import modin.pandas as pd
```

pandas API

MODIN

Ray  Dask  ...

https://github.com/modin-project/modin

- 10+ Million Downloads (1M+/month)
- 100+ contributors
- Used by 40+ companies & orgs
- Part of AWS, Intel distros

“Core” Dataframe Algebra
Parallelization Framework
The Next Generation with Ponder:
Data Science at scale now powered by your database
The Next Generation with Ponder: Data Science at scale now powered by your database
The Magic of Ponder

Min-Max normalization

\[ x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)} \]

One-line in pandas

```python
(df - df.min()) / (df.max() - df.min())
```
The Magic of Ponder

Min-Max normalization

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

One-line in pandas

$$(df - df.min()) / (df.max() - df.min())$$

Equivalent SQL is 300+ lines!!
Easy to Get Started

Create a database connection

```python
import duckdb
db_con = duckdb.connect()
```

Start running pandas directly in your database

```python
import ponder
ponder.init()
import modin.pandas as pd
df = pd.read_sql("DB_TABLENAME", con=db_con)

df.describe()
df.groupby("...").mean()
pd.concat([df, df]) # ... and much more!
```
API-level Performance – Ponder on DuckDB vs. Pandas

Ponder on DuckDB outperforms pandas when scaling to large datasets.
Two Minutes to Scalable Pandas with DuckDB!

1. Create an account: [QR Code](http://app.ponder.io/signup)

2. Install Ponder on your machine

   `pip install ponder`

3. Log into Ponder

   `ponder login`
Backup Slides
With pandas

- Fetch data with SQL

  - Limited samples
  - Out-of-memory errors
  - Expensive I/O costs
  - Un-secure access

With PONDER

- Let your database do the heavy lifting

  - Ready for production
  - Leverage database optimizations
  - No infrastructure setup required
  - Securely access data where it lives
It is difficult to map Pandas to SQL, while supporting:

- **Order Semantics**
- **Metadata** (mixed types, data/metadata equivalence)
- **Incremental** query construction
- **Coverage** of 600+ APIs (incl. ops not possible in SQL systems)

Modin instead maps to distrib. compute engines (execute anything they’re told).