

DuckDB @ DataCamp

Embedding DuckDB in Applications



Hello!

👋 I'm Rik

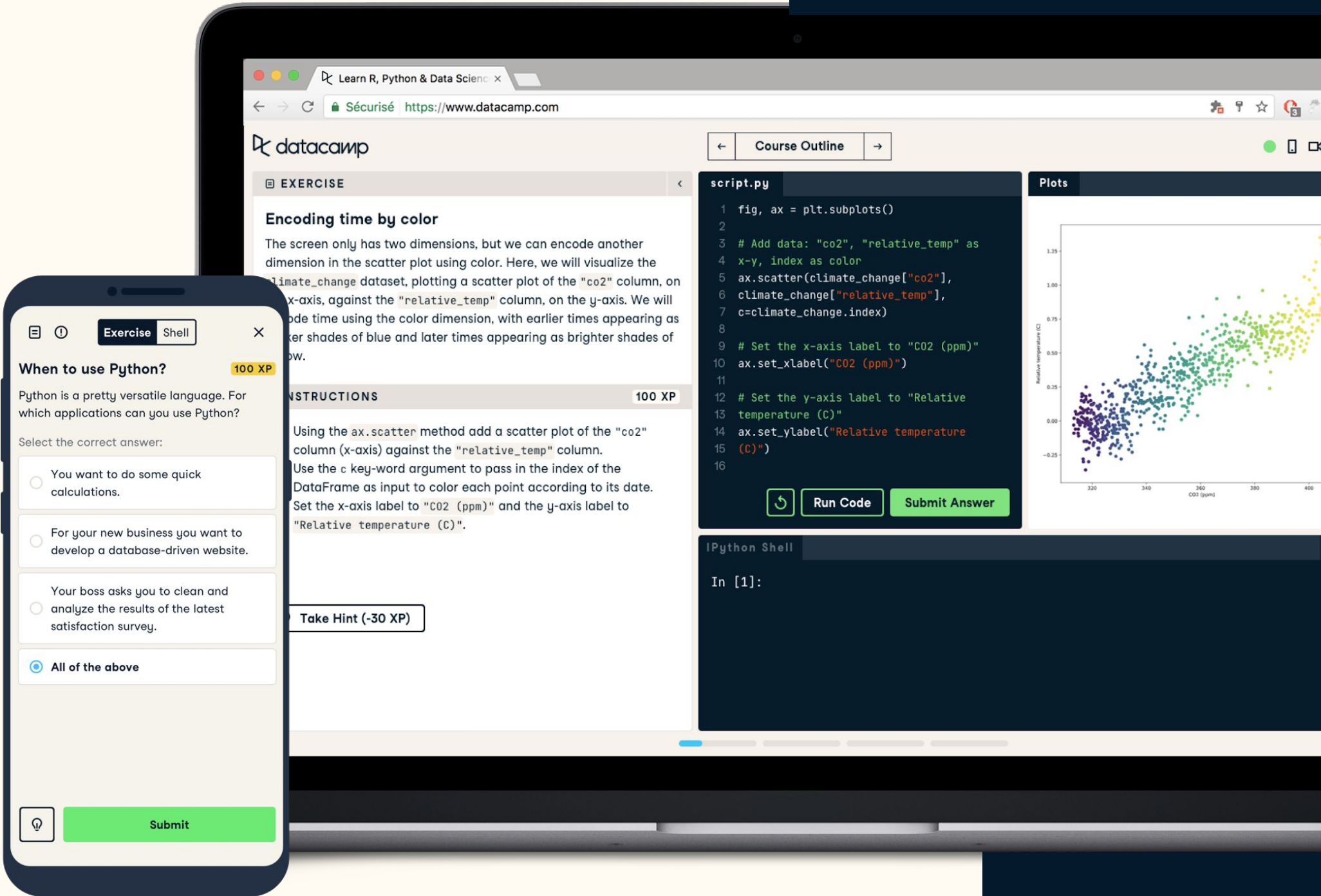
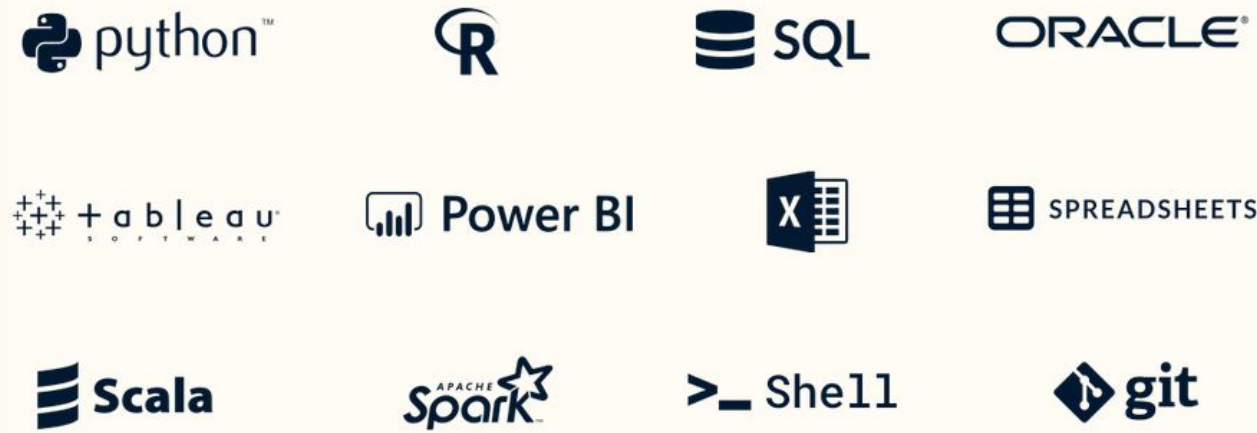


Rik Bauwens

Software Engineer @ DataCamp



Hands-on data skill building



What does DataCamp do?

1. Teach Data Skills

- a. Python, R, Julia, ...
- b. SQL
- c. BI
- d. Certifications
- e. Media
- f. ...

2. Create Data Tooling

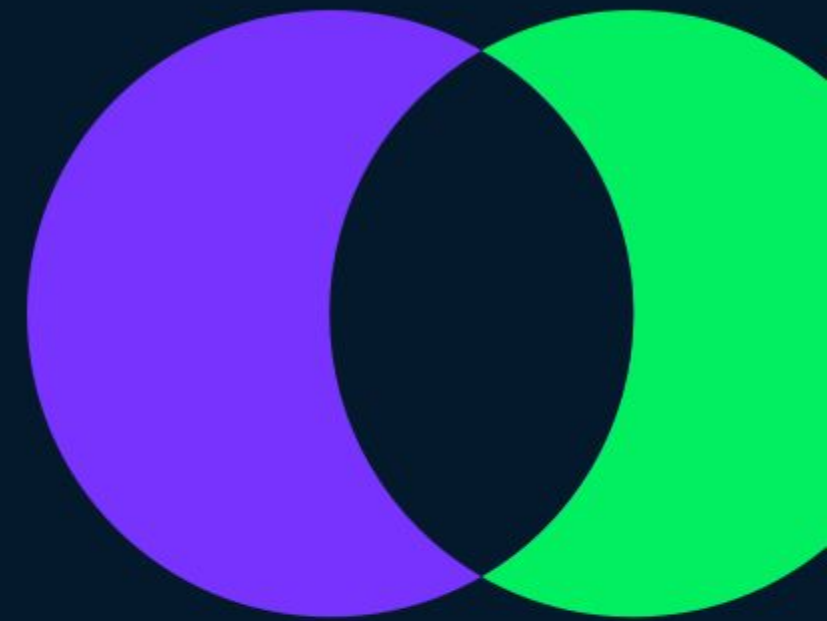
- a. "Workspace", our notebook product





1

The Notebook




```
[5]: import matplotlib.pyplot as plt
plt.style.use('classic')
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
sns.set()
```

```
[6]: rng = np.random.RandomState(0)
x = np.linspace(0, 10, 500)
y = np.cumsum(rng.randn(500, 6), 0)
```

Next step

Now, create a graph.

```
[7]: plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
```






DataCamp Workspace

A modern, cloud-based data
science notebook



Revenue Metrics

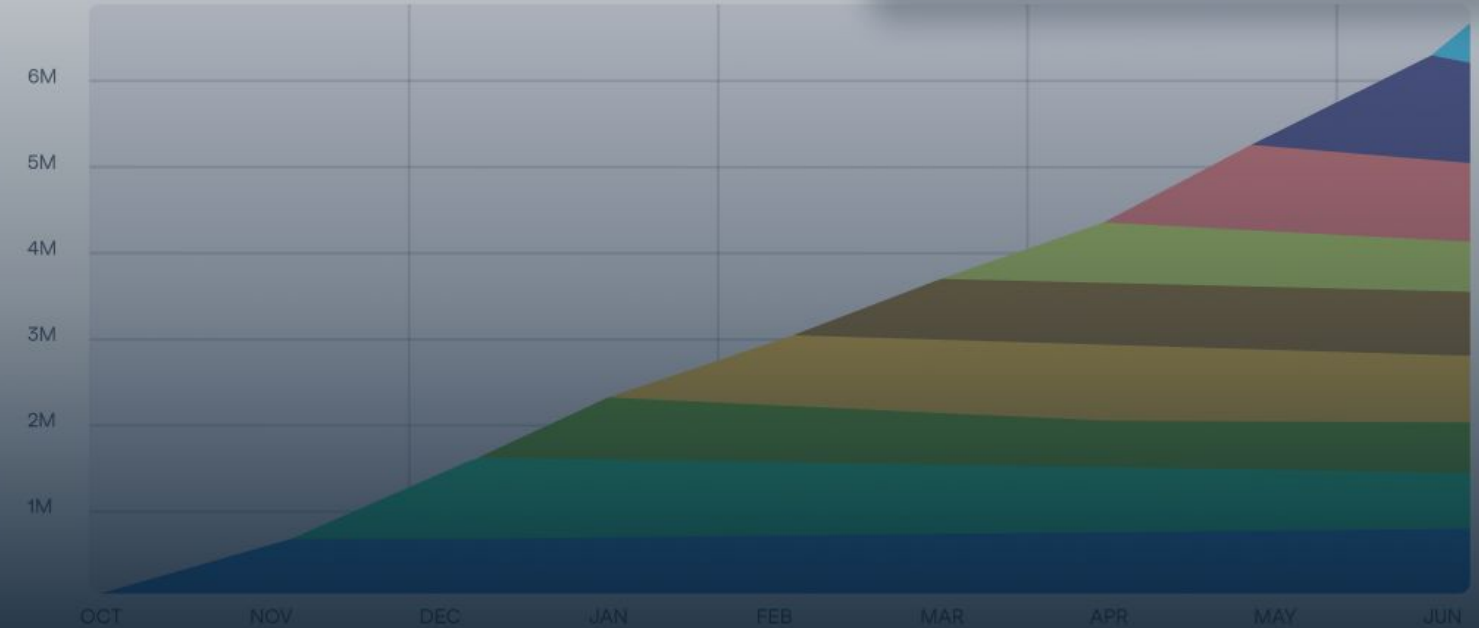
 | Tell our AI what to do...


acme-prod ▾ DataFrame ▾ available as df1  

SELECT
subscribed_month,
payment_month,
SUM(revenue) AS revenue
FROM views.mart_b2c_daily_cohorts
GROUP BY 1, 2


Create layer-cake graph
fig = px.area(
df_month,
x='payment_month',
y='revenue',
color="subscribed_month",
)

Revenue by Monthly Cohort



 **Jack Defoe**
Yesterday, 09:13 AM

Hey Rosaline, check out this breakdown of B2C revenue by monthly cohort 🙄

 **Rosaline Ropuiz**
Today, 10:30 AM

This is great! 😊 Can you provide a similar breakdown for weekly cohorts in the past 3 months?

Write your comment here...



Ticket Sales ▾

DataFrame ▾

available as tickets



```
1 SELECT
2     eventname,
3     catname,
4     venuecity,
5     caldate
6 FROM event
7     INNER JOIN date USING(dateid)
8     INNER JOIN category USING(catid)
9     INNER JOIN venue USING(venueid)
10 ORDER BY caldate
11 LIMIT 5
```

[1] ●

	eventname ▾	catname ▾	venuecity ▾	caldate ▾
0	Grease	Musicals	Boston	2008-01-01T00:00:00.000Z
1	The King and I	Musicals	New York City	2008-01-01T00:00:00.000Z
2	Return To Forever	Pop	Newark	2008-01-01T00:00:00.000Z
3	Gretchen Wilson	Pop	Seattle	2008-01-01T00:00:00.000Z
4	Nas	Pop	Boston	2008-01-01T00:00:00.000Z

Table

Chart

5 rows ↓


```
import pandas as pd
```

[3] ●

```
df = pd.read_parquet('data/data.parquet')
df.head()
```

	vendor_id ▾	pickup_at ▾	dropoff_at ▾	passenger_count ▾	trip_distance ▾	pickup_longitude ▾	pickup_latitude ▾
0	VT	2010-12-03 00:22:00.000000...	2010-12-03 00:30:00.000000...	1	1.99	-73.982185	40.761432
1	VT	2010-12-05 01:37:00.000000...	2010-12-05 01:41:00.000000...	1	0.78	-73.9864	40.761432
2	VT	2010-12-01 12:14:00.000000...	2010-12-01 12:36:00.000000...	2	2.06	-73.99695	40.761432
3	VT	2010-12-05 06:44:00.000000...	2010-12-05 06:58:00.000000...	1	5.03	-73.98933	40.761432
4	VT	2010-12-03 08:28:00.000000...	2010-12-03 08:51:00.000000...	1	2.61	-73.95514	40.761432

Table Chart

5 rows ↓

DataFrames and CSVs ▾

DataFrame ▾ available as df

```
SELECT *
FROM df
WHERE passenger_count > 3
ORDER BY passenger_count ASC
LIMIT 5
```

[4] ●

	vendor_id ▾	pickup_at ▾	dropoff_at ▾	passenger_count ▾	trip_distance ▾	pickup_longitude ▾	pickup_latitude ▾
0	VT	2010-12-01 15:26:00.000000...	2010-12-01 15:35:00.000000...	4	0.4	-73.98557	40.761432
1	VT	2010-12-04 21:10:00.000000...	2010-12-04 21:18:00.000000...	4	1.81	-73.97517	40.761432
2	CMT	2010-12-04 22:43:53.000000...	2010-12-04 22:58:43.000000...	4	1.5	-73.98538	40.761432
3	VT	2010-12-04 02:11:00.000000...	2010-12-04 02:18:00.000000...	4	1.28	-73.98835	40.761432
4	VT	2010-12-04 11:14:00.000000...	2010-12-04 11:27:00.000000...	4	1.33	-73.98559	40.761432

Table Chart

5 rows ↓

```
import duckdb
variable_name = duckdb.sql(query).df()
variable_name
```

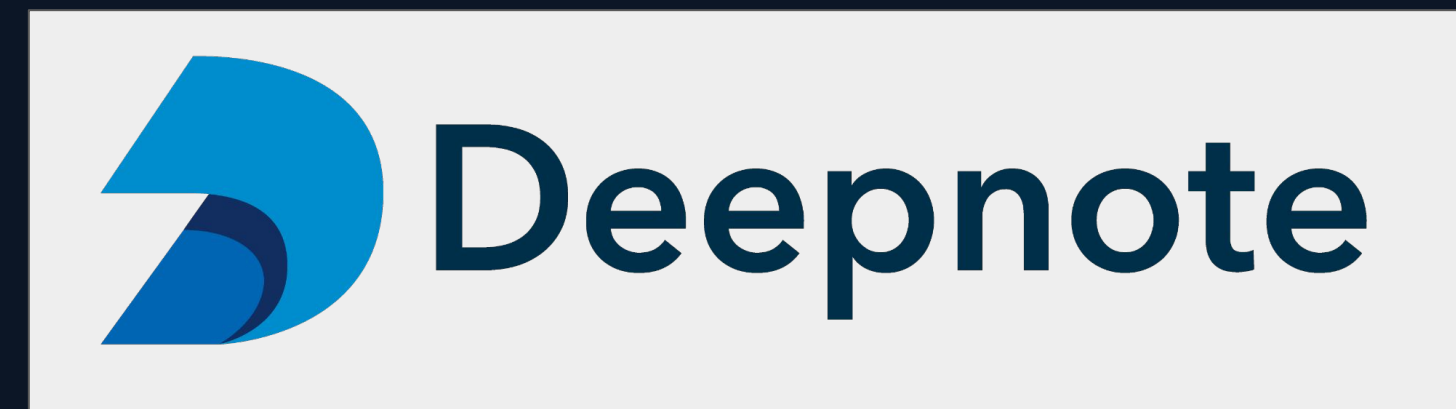
DuckDB @ DataCamp

We're not the only notebook that embeds DuckDB...



↳ `adelie_penguins`

Dataframe SQL queries execute right in Hex using DuckDB, so the SQL syntax is very similar to PostgreSQL, and the full reference of



```
FROM 'path/to/my_data.csv'
```

 This uses duckdb under the hood. Visit the [duckdb reference](#) for a specific flavor of SQL.

+ Block

+ Code % + J

+  digital ocean postgres Query

+  dba Query



Adrien Duchateau / Time Series Decomposition

File Edit View Insert Run Help Last edit was seconds ago

Run Share Reader mode

2 hidden cells

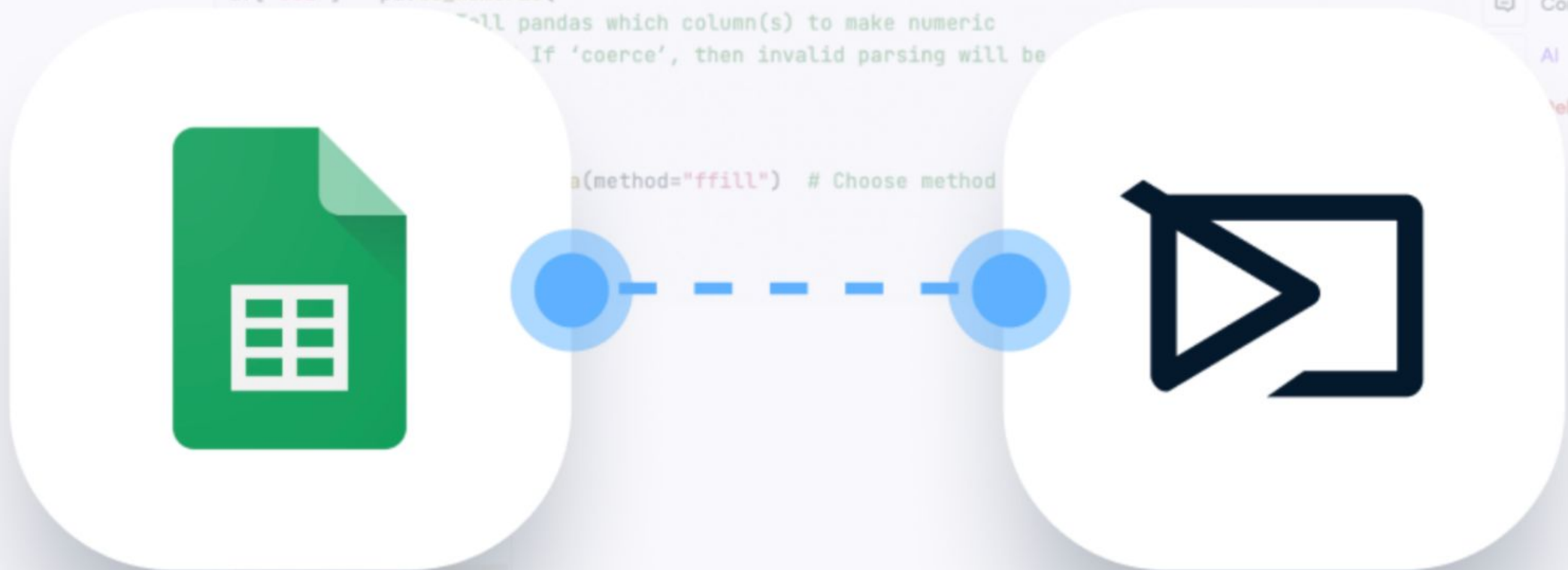
Convert numeric columns
df["co2"] = pd.to_numeric(
Tell pandas which column(s) to make numeric
If 'coerce', then invalid parsing will be
a(method="ffill") # Choose method

1958-04-19 317.5
1958-04-26 316.4

Plot settings
%config InlineBackend.figure_format='retina'
plt.style.use("ggplot")
plt.rcParams["figure.figsize"] = 12, 9 # Figure size (width,height)

Perform time series decompositon
decomposition = sm.tsa.seasonal_decompose(df)

Run Comment AI Delete



DuckDB @ DataCamp

Google Sheets SQL

→ Sheets often contain multiple “logical” tables

	A	B	C	D	E	F	G
1							
2		City	Country		Country	Good Beer	
3		Amsterdam	Netherlands		Netherlands	No	
4		Rotterdam	Netherlands		Belgium	Yes	
5		Antwerp	Belgium				
6		Brussels	Belgium				
7							



	A	B	C	D	E	F	G
1		countries			facts		
2		City	Country		Country	Good Beer	
3		Amsterdam	Netherlands		Netherlands	No	
4		Rotterdam	Netherlands		Belgium	Yes	
5		Antwerp	Belgium				
6		Brussels	Belgium				
7							

```

SELECT *
FROM 'Sheet1!B2:C6' AS countries
JOIN 'Sheet1!E2:F4' AS facts
ON countries.country = facts.country

```



	A	B	C	D	E	F	G
1							
2		City	Country		Country	Good Beer	
3		Amsterdam	Netherlands		Netherlands	No	
4		Rotterdam	Netherlands		Belgium	Yes	
5		Antwerp	Belgium				
6		Brussels	Belgium				
7							

1. Parse SQL

```

SELECT *
FROM 'Sheet1!B2:C6' AS countries
JOIN 'Sheet1!E2:F4' AS facts
ON countries.country = facts.country

```



	A	B	C	D	E	F	G
1							
2		City	Country		Country	Good Beer	
3		Amsterdam	Netherlands		Netherlands	No	
4		Rotterdam	Netherlands		Belgium	Yes	
5		Antwerp	Belgium				
6		Brussels	Belgium				
7							

2. Download Ranges



```

SELECT *
FROM 'Sheet1!B2:C6' AS countries
JOIN 'Sheet1!E2:F4' AS facts
ON countries.country = facts.country

```



	A	B	C	D	E	F	G
1							
2		City	Country		Country	Good Beer	
3		Amsterdam	Netherlands		Netherlands	No	
4		Rotterdam	Netherlands		Belgium	Yes	
5		Antwerp	Belgium				
6		Brussels	Belgium				
7							

```

SELECT *
FROM 'Sheet1!B2:C6' AS countries
JOIN 'Sheet1!E2:F4' AS facts
ON countries.country = facts.country


```



3. Register with DuckDB



	A	B	C	D	E	F	G
1							
2		City	Country		Country	Good Beer	
3		Amsterdam	Netherlands		Netherlands	No	
4		Rotterdam	Netherlands		Belgium	Yes	
5		Antwerp	Belgium				
6		Brussels	Belgium				
7							


Country Data ▾

DataFrame ▾
available as df1

⌵
🔗

```

SELECT *
FROM 'Sheet1!B2:C6' AS countries
JOIN 'Sheet1!E2:F4' AS facts
ON countries.country = facts.country

```

[7] ●

	City ▾	Country ▾	Good Beer ▾
0	Amsterdam	Netherlands	No
1	Rotterdam	Netherlands	No
2	Antwerp	Belgium	Yes
3	Brussels	Belgium	Yes

Table
Chart

4 rows
⌵



DuckDB @ DataCamp

Our experience embedding DuckDB

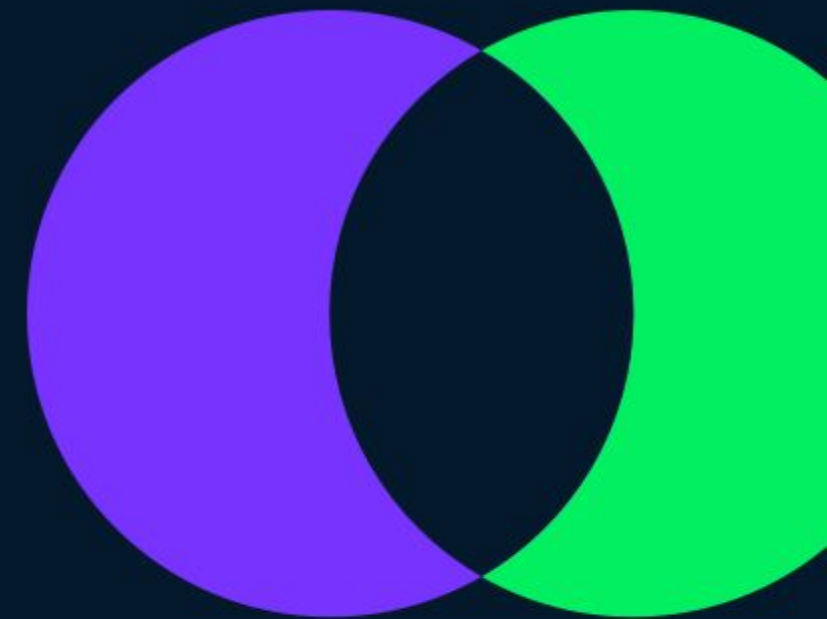
- Great DX
 - ◆ Simple to embed
- Feature complete
 - ◆ Parquet, CSV, Apache Arrow, ... support
- Portable
 - ◆ We run DuckDB in Python & on Node





2

Teaching SQL



USING in action | SQL

campus.datacamp.com/courses/joining-data-in-sql/introducing-inner-joins?ex=4

Learn / Courses / Joining Data in SQL

Course Outline

Daily XP 100

Exercise

USING in action

In the previous exercises, you performed your joins using the `ON` keyword. Recall that when **both** the field names being joined on are the same, you can take advantage of the `USING` clause.

You'll now explore the `languages` table from our database. Which languages are official languages, and which ones are unofficial?

You'll employ `USING` to simplify your query as you explore this question.

Instructions

100 XP

- Use the country `code` field to complete the `INNER JOIN` with `USING` ; do **not** change any alias names.

Take Hint (-30 XP)

query.sql

Light Mode

```
1 SELECT c.name AS country, l.name AS language, official
2 FROM countries AS c
3 INNER JOIN languages AS l
4 USING (code)
```

↶

Run Code

Submit Answer

query result

languages

countries

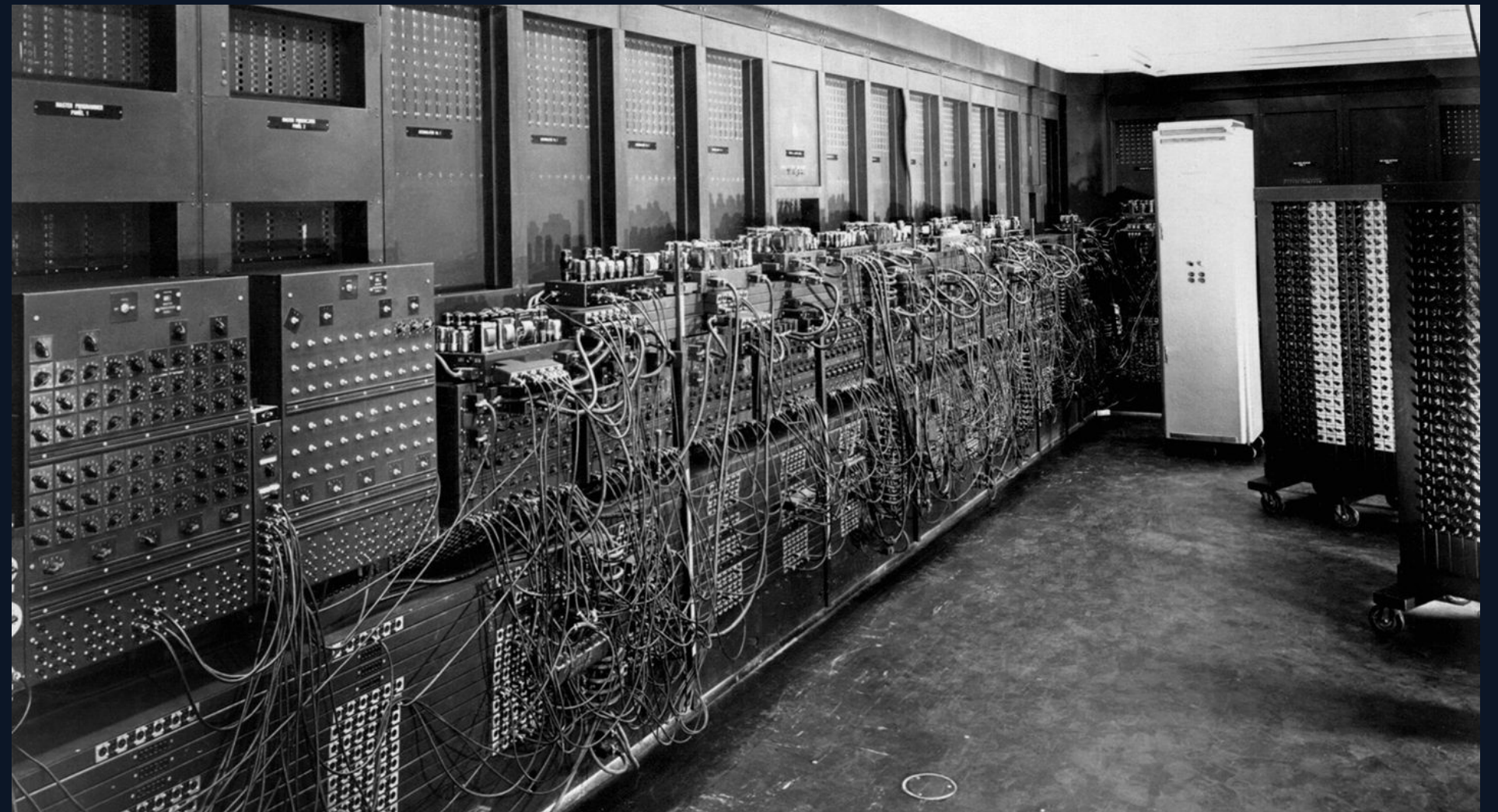
Belarus	Russian	true
Belarus	Belarusian	true
Belarus	Other	false
Belarus	unspecified	false
Belgium	Dutch	true

Showing 100 out of 909 rows

DuckDB @ DataCamp

Teaching SQL

- Can't ask users to run a database locally
- SQL is executed on the server
- But now there's WebAssembly...



DuckDB @ DataCamp

Teaching SQL

- Evaluating running code in the browser with Wasm
- DuckDB-Wasm for SQL courses!
 - ◆ SQL among the most popular courses
 - ◆ Bundle size is the biggest blocker
 - ◆ Love the feature completeness & performance



Thank you

