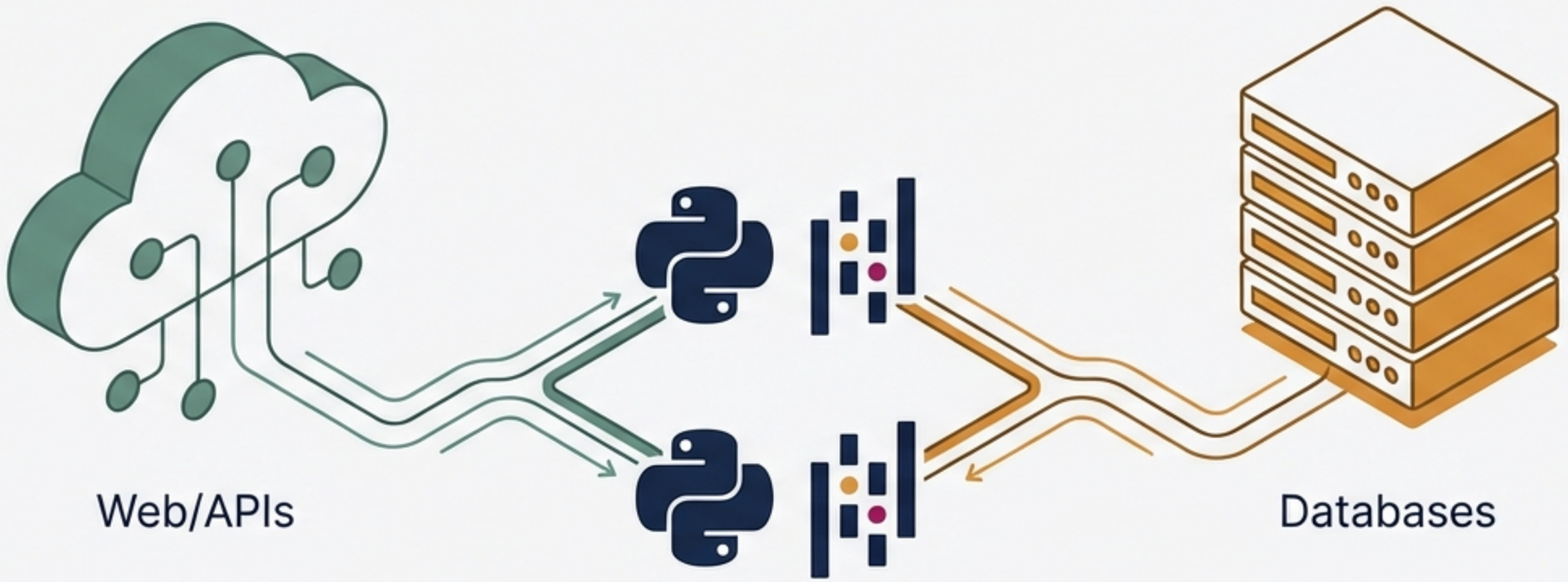
Data Ingestion with Python: JSON & SQL

Moving Beyond the CSV: Mastering APIs and Relational Databases with Pandas



Based on the 100 Days of Machine Learning Curriculum - Day 16

Real-World Data Doesn't Always Come in a CSV

While beginners start with Comma Separated Values, production-grade data pipelines require handling more complex structures.

CSV

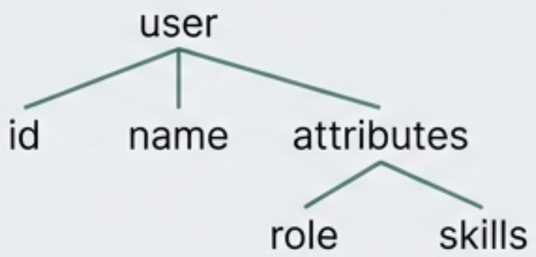
Static & Flat

ID	Name	Role
1	Alice	Eng
2	Bob	Des

Static & Flat

JSON

Nested & Web-Ready

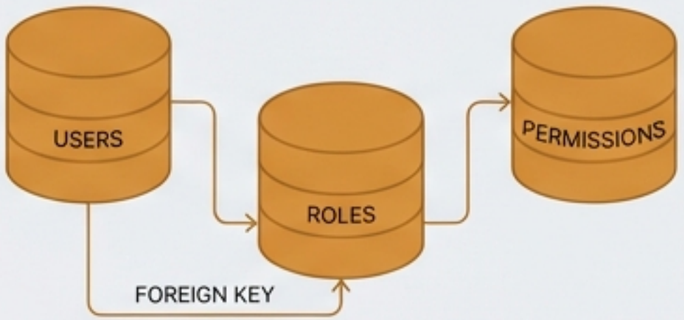


Example Payload

```
{
  "user": {
    "id": 123,
    "name": "Charlie",
    "attributes": {
      "role": "Dev",
      "skills": ["Python", "SQL"]
    }
}
```

SQL

Structured & Scalable

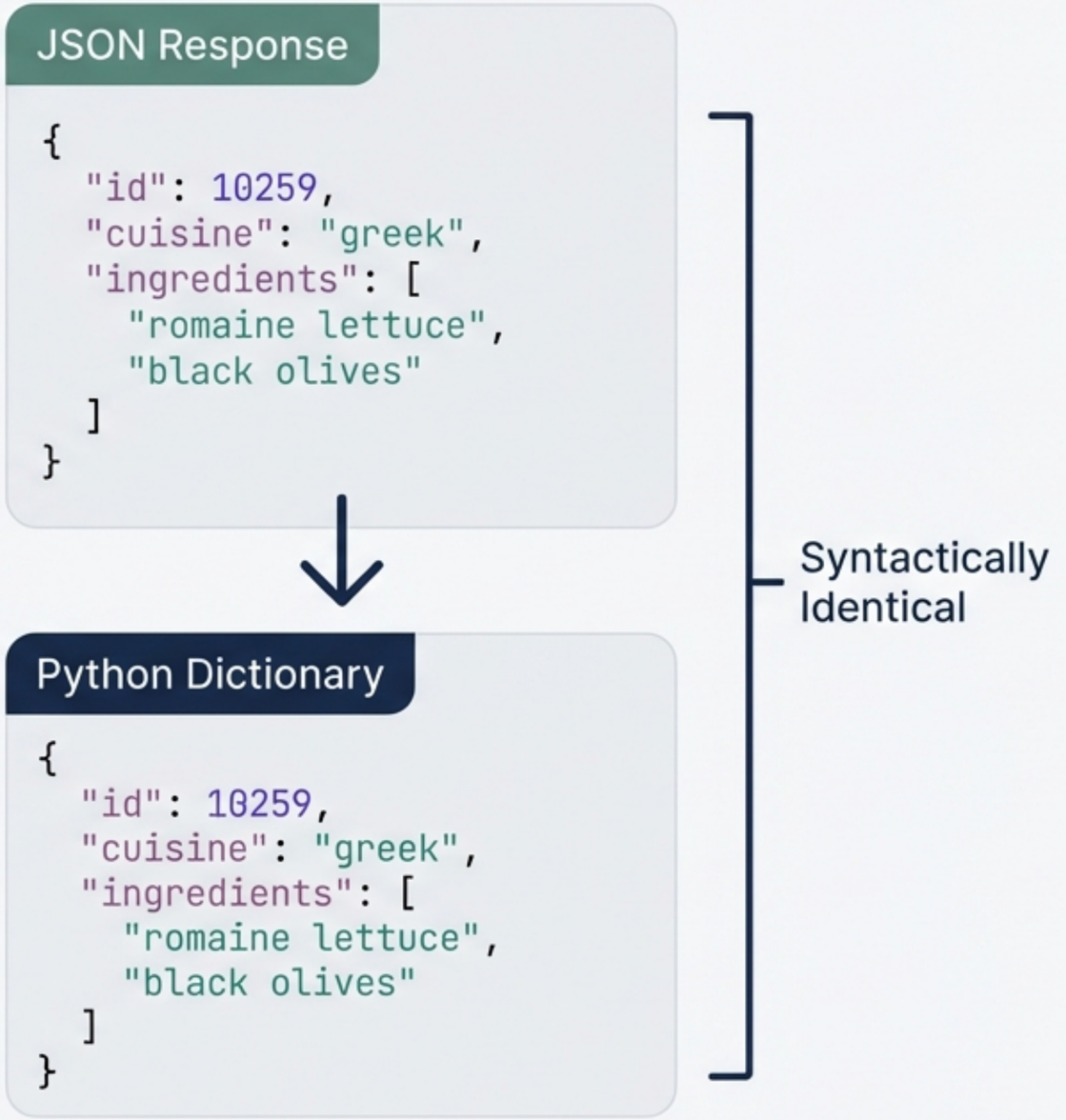


Relational Query

```
SELECT
  u.name,
  r.role_name
FROM users u
JOIN roles r ON u.role_id = r.id
WHERE u.active = TRUE;
```

Module 1: The Universal Format (JSON)

JSON (JavaScript Object Notation) is a universal data format based on key-value pairs. It is the standard language of Web APIs; when a browser requests data, the response is almost always in JSON.



Ingesting Local JSON Files

pd.read_json() automatically detects key-value structures and flattens them.

```
import pandas as pd

# Reading a local file containing cuisine data
df = pd.read_json('train.json')
df.head()
```

id	cuisine	ingredients
10259	greek	['romaine lettuce', 'black olives', ...]
25693	southern_us	['plain flour', 'ground pepper', ...]
20130	filipino	['eggs', 'pepper', 'salt', ...]

Fetching Live Data from URLs

```
# Fetching currency exchange rates directly
url = 'https://api.exchangerate-api.com/v4/latest/INR'
df = pd.read_json(url)
```

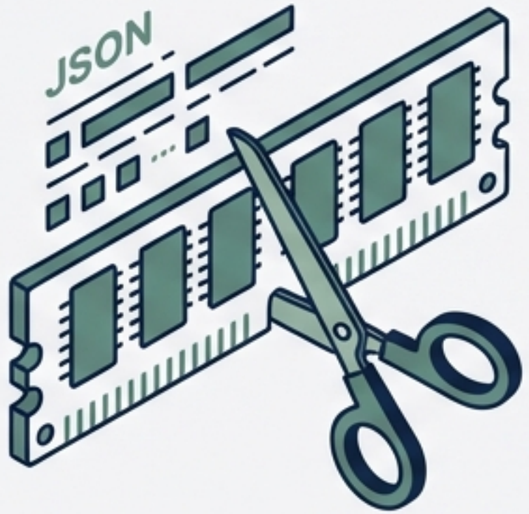
Live Request

For automated pipelines, Pandas can read directly from an API endpoint without manual downloading.

Exchange Rate Data

provider	base	date	rates
https://www.exchangerate-api.com	INR	2024-05-20	{'USD': 0.012, 'EUR': 0.011, 'GBP': 0.009...}

Optimising JSON Ingestion



Chunking

For massive JSON files, use the 'chunksize' parameter to load data in parts. Prevents RAM overflow.

Date Handling

Use 'convert_dates' parameters to ensure time-series strings are parsed into datetime objects immediately.

Encoding

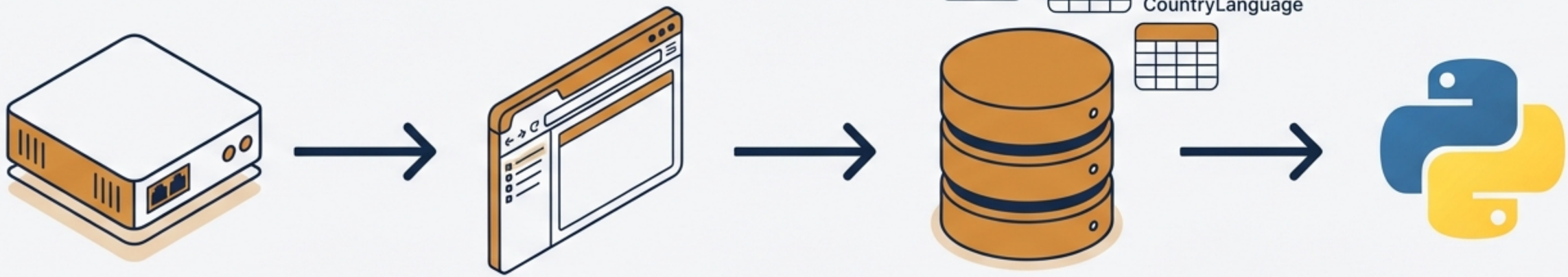
If special characters appear corrupted, adjust the 'encoding' parameter (e.g., 'utf-8').

Module 2: The Enterprise Bridge (SQL)

Extracting data directly from databases without intermediate files.

XAMPP / MySQL

phpMyAdmin



XAMPP / MySQL

The Source

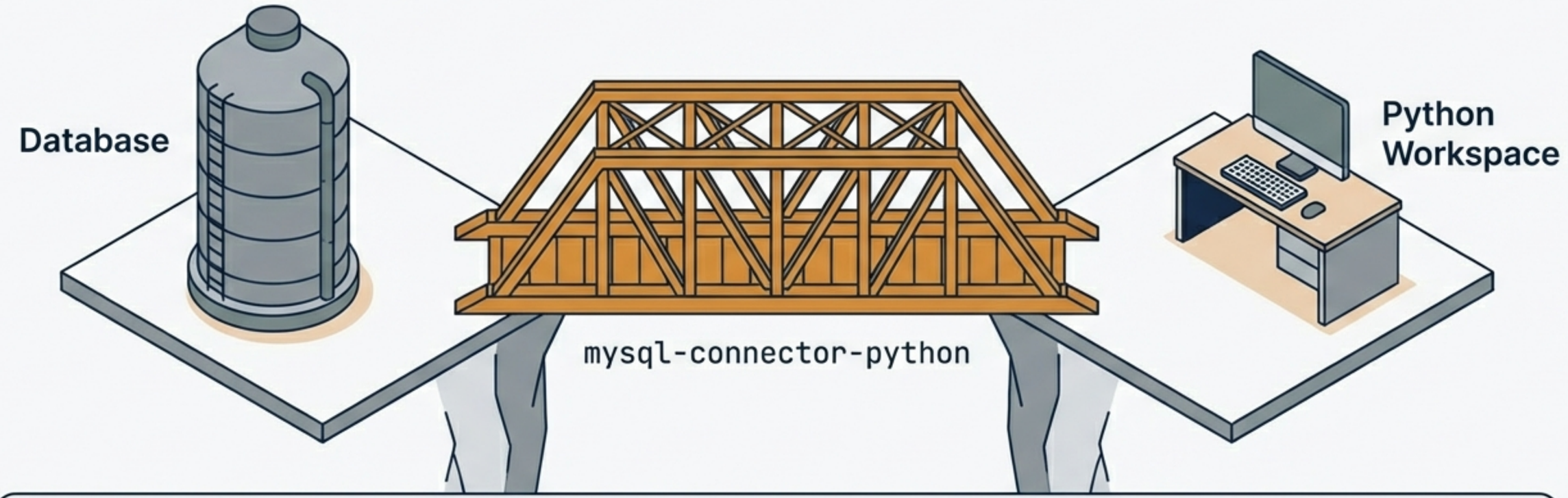
phpMyAdmin

The Interface

World DB

Installing the Connector

Python requires a specific driver to communicate with MySQL databases.

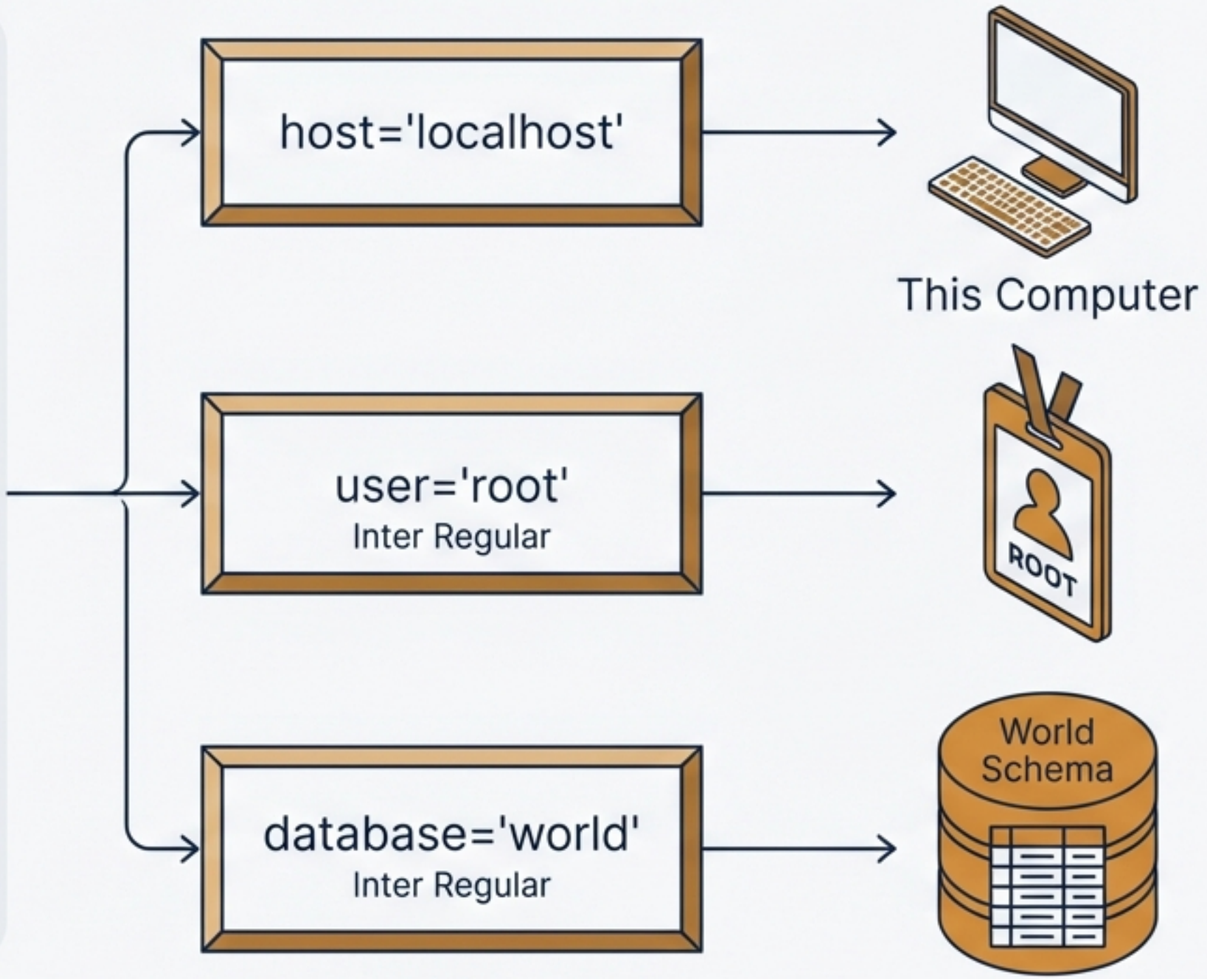



```
!pip install mysql-connector-python
```

Establishing the Connection

```
import mysql.connector

# Creating the connection object
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='',
    database='world'
)
```



The Magic Function: read_sql_query

pd.read_sql_query(query, connection)

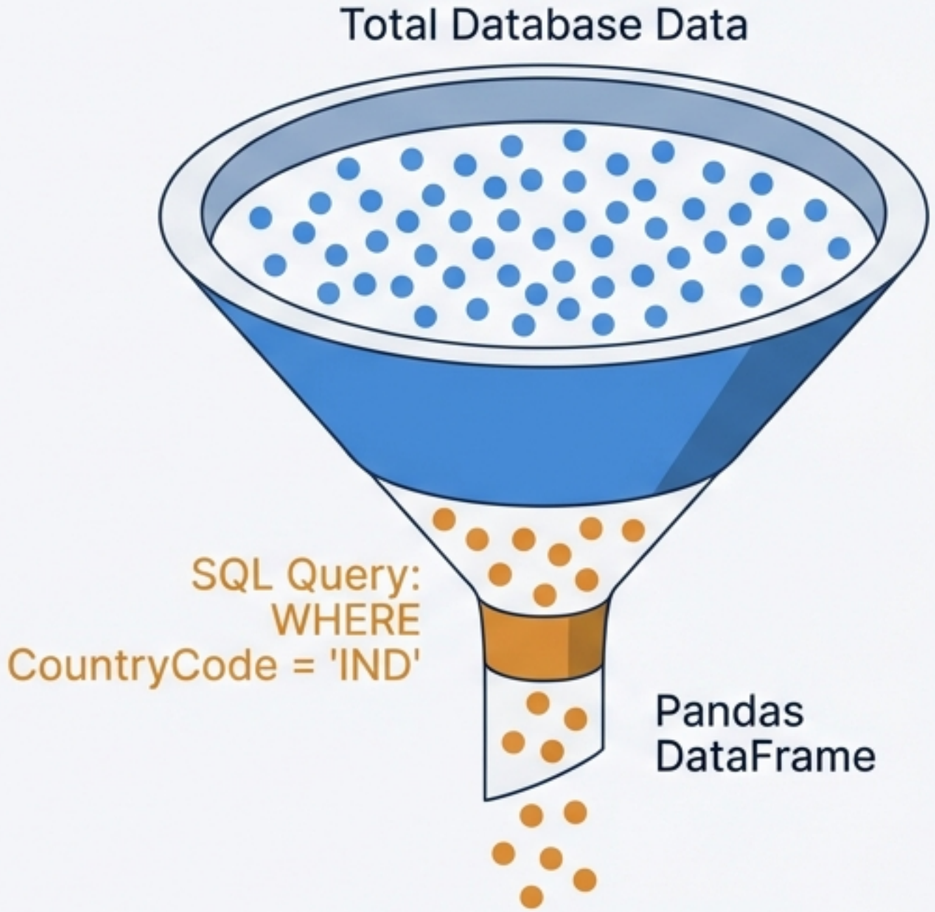
```
# Extracting the entire 'city' table
df = pd.read_sql_query("SELECT *
FROM city", conn)
df.head()
```

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800

The Power of Filtering (The Funnel)

Don't load the whole database. Use SQL WHERE clauses to filter data before it hits Python's memory.

```
# filtering for Indian cities only
query = "SELECT * FROM city WHERE CountryCode = 'IND'"
df_india = pd.read_sql_query(query, conn)
```



Name	CountryCode
Mumbai	IND
Delhi	IND
Madras	IND

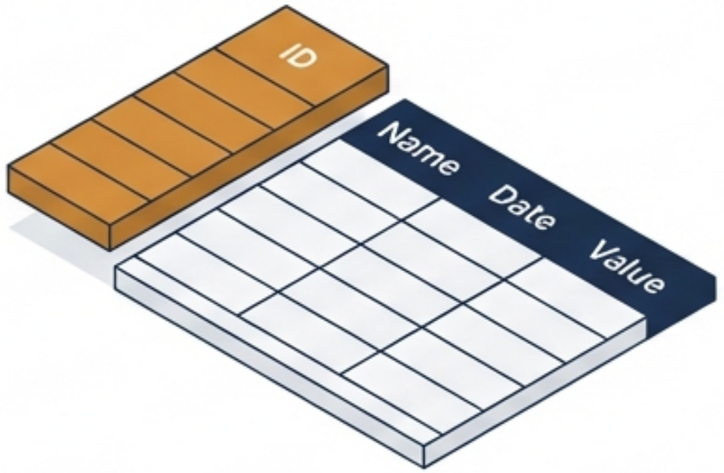
Advanced Logic Extraction

SQL can handle complex logic, offloading computational work to the server.

```
# Fetching countries with high life expectancy
query = "SELECT * FROM country WHERE LifeExpectancy > 60"
df_life = pd.read_sql_query(query, conn)
```

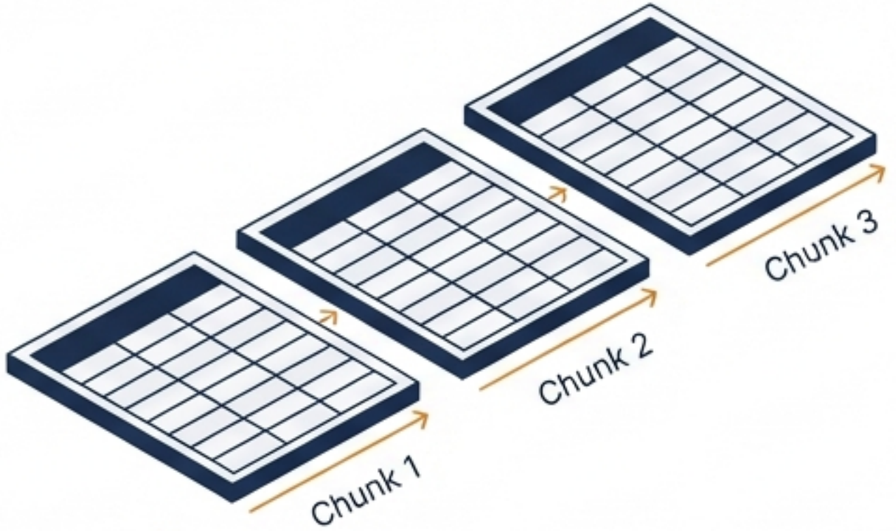
	Code	Name	Continent	LifeExpectancy
1	ABW	Aruba	North America	78.4
2	AGO	Angola	Africa	38.3 ✕
3	AIA	Anguilla	North America	76.1

SQL Ingestion Best Practices



index_col

Specify a column to become the DataFrame index immediately upon loading.



chunksize

Process very large tables in batches to save memory.

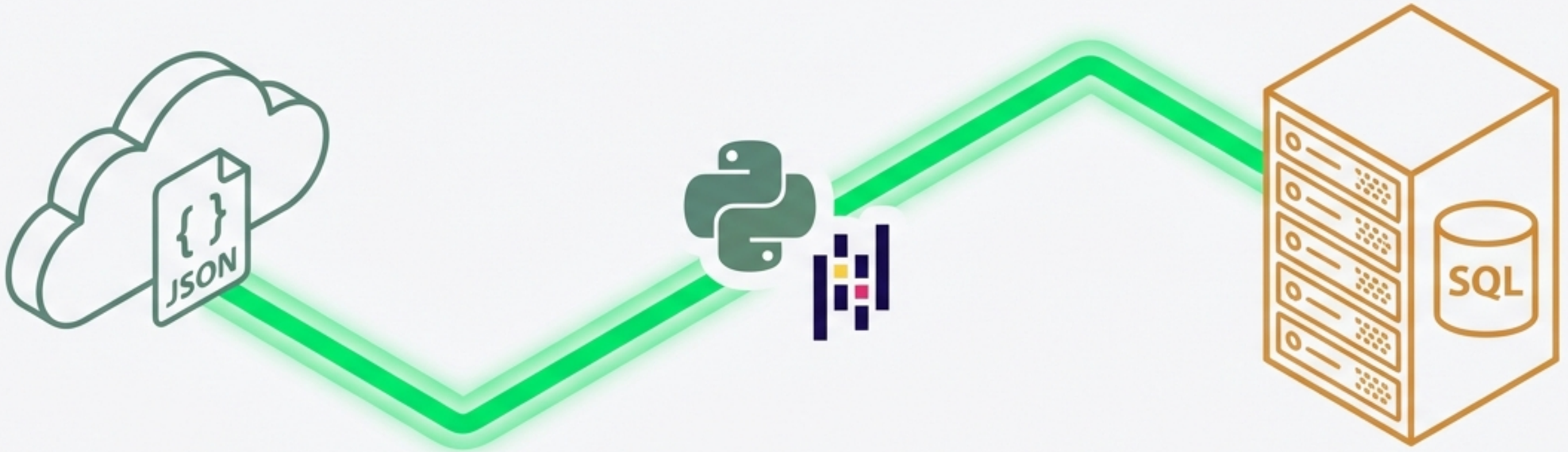
```
Always remember: conn.close()
```

Syntax Cheat Sheet

JSON		SQL	
Function:	`pd.read_json()`	**Function:**	`pd.read_sql_query()`
Input:	File path or URL	**Input:**	SQL Query String + Connection Object
Best For:	API responses, nested web data	**Best For:**	Relational databases, large structured datasets

Your Data Toolkit is Expanded

You can now bridge the gap between Python and the two vast reservoirs of world data: The Web (JSON) and The Enterprise (SQL).



Next: Building custom datasets from live APIs.