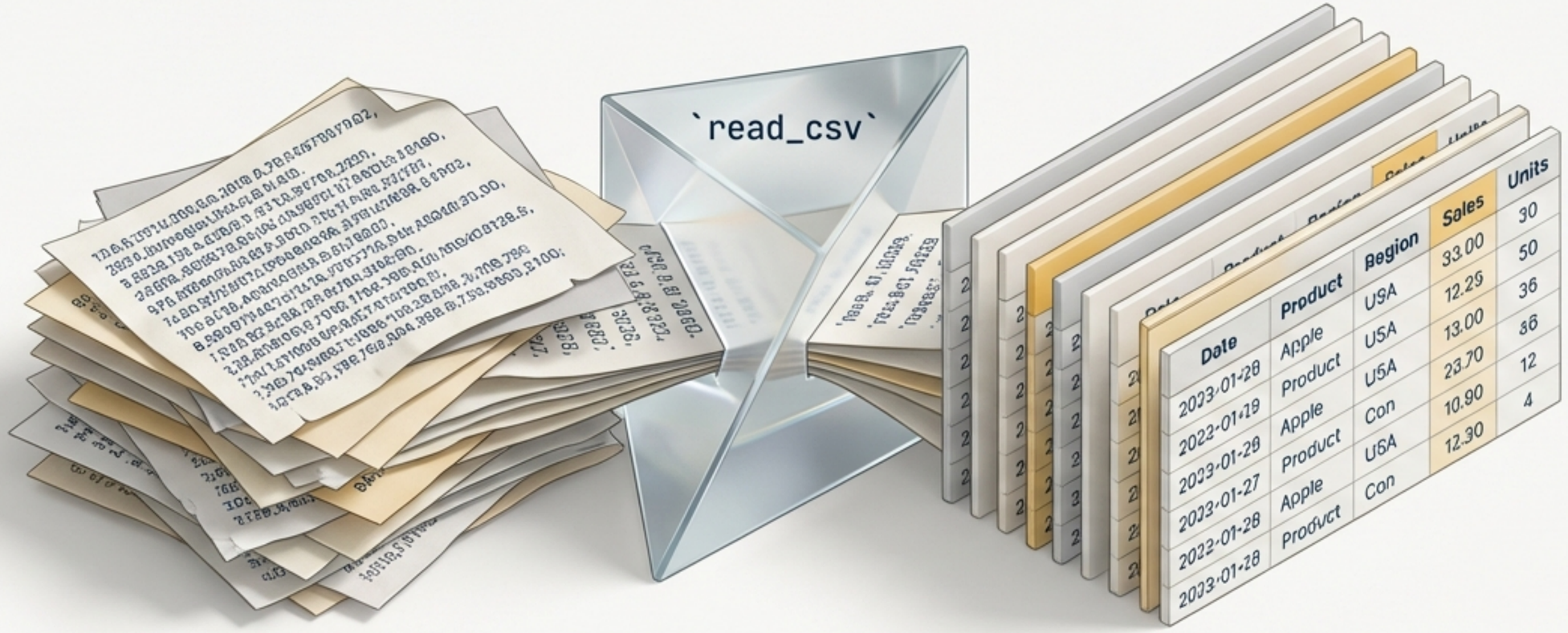
Mastering CSV Data in Pandas

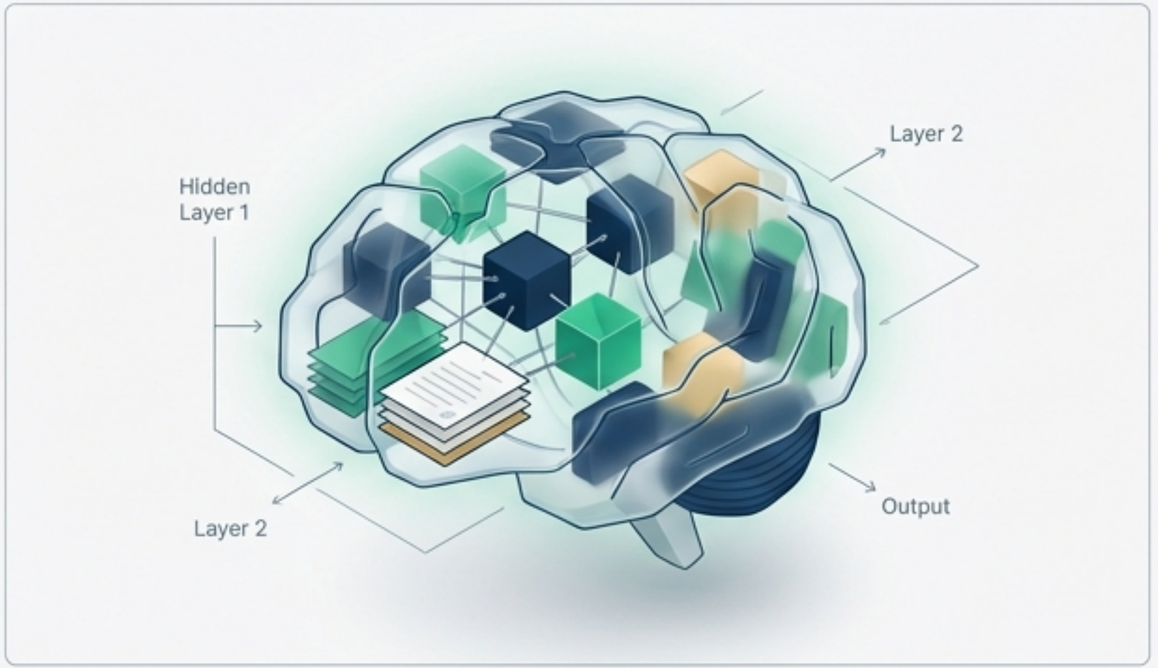
From Basic Ingestion to Advanced Data Engineering



Day 15 | 100 Days of Machine Learning | A guide to the read_csv function parameters

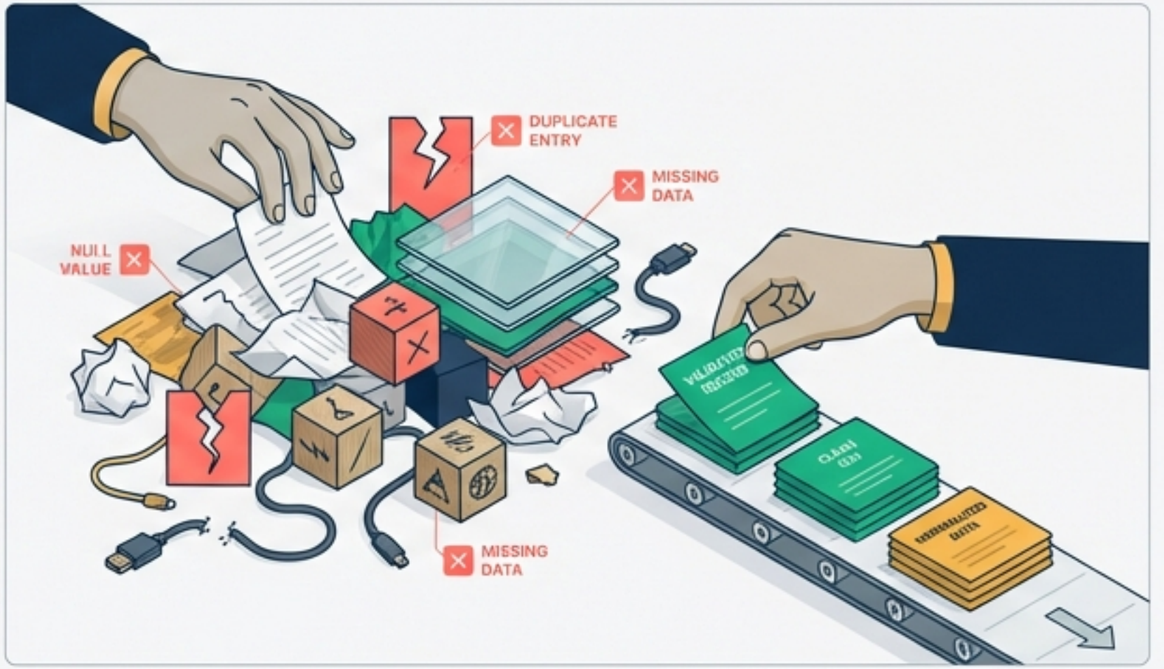
If you fix the data input, the algorithm performs.

The Expectation: 10% Modelling



Perceived focus on complex architectures and hyperparameter tuning.

The Reality: 90% Data Cleaning



Actual time spent on preprocessing, validation, and formatting.

“If you feed a good algorithm bad data, performance suffers. If you feed a bad algorithm good data, it has a fighting chance.”

Before JSON or SQL, mastering the Comma Separated Value is the first step in the data science journey.

The Universal Opener -20

Loading a clean file from a local drive

```
import pandas as pd
df = pd.read_csv('aug_train.csv')
df
```

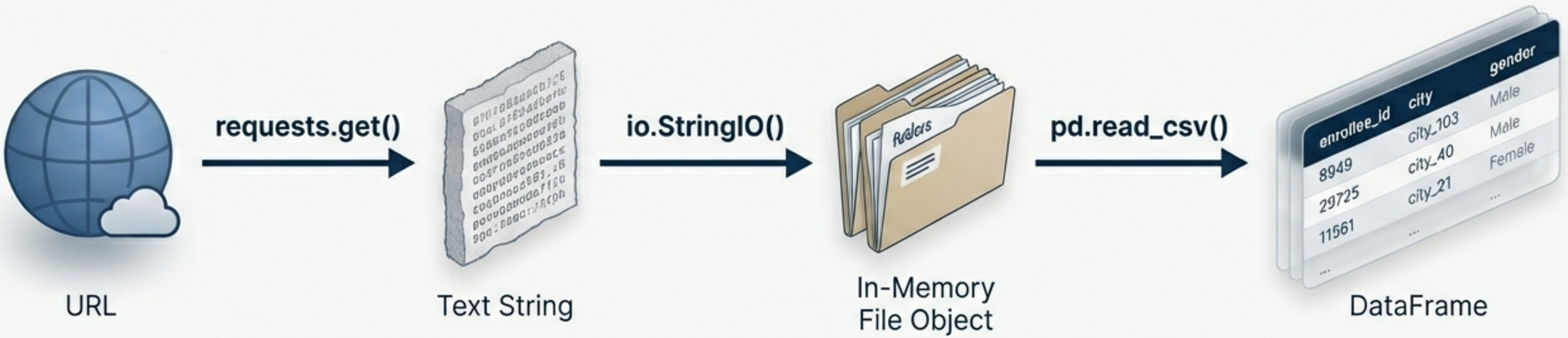
Path to local file

enrollee_id	city	gender
8949	city_103	Male
29725	city_40	Male
11561	city_21	Female
...	...	...

Default assumption: Comma separator,
First row is header

Fetching Data from Remote Servers

Using requests and StringIO to simulate a file object.



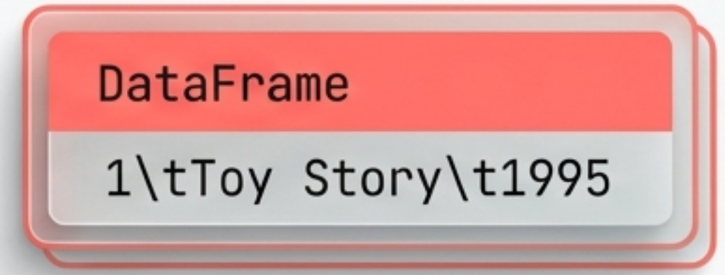

```
import requests
from io import StringIO

url = "https://raw.githubusercontent.com/..."
req = requests.get(url)
data = StringIO(req.text)

df = pd.read_csv(data)
```

When Commas Aren’t Commas

Handling Tab Separated Values (TSV).

Default Load (Fails) Inter Regular

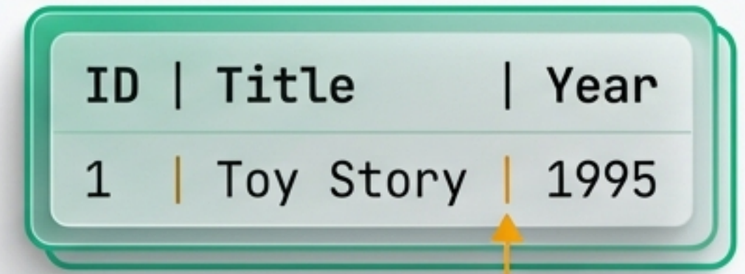
```
pd.read_csv('movie_titles.tsv')
```



All data mashed into one column, separated by tabs.

The Fix: Define Separator Inter Regular

```
pd.read_csv('movie_titles.tsv', sep='\t')
```



Cleanly separated into columns using the specified delimiter.

The Header Headache

Inter Regular, slidlins side-side notc: mattic plastic table rows, and the last headarthis with diffused shadows.

No Header in File

```
pd.read_csv(..., names=['First','Last','Age'])
```

	First	Last	Age
1	John	Doe	30
2	Ray	Coury	45
4	John	Doe	30
5	Ray	Coury	45
6	John	Doe	30

Data without a header. The `names` argument supplies column titles.

Header Buried in Row 2

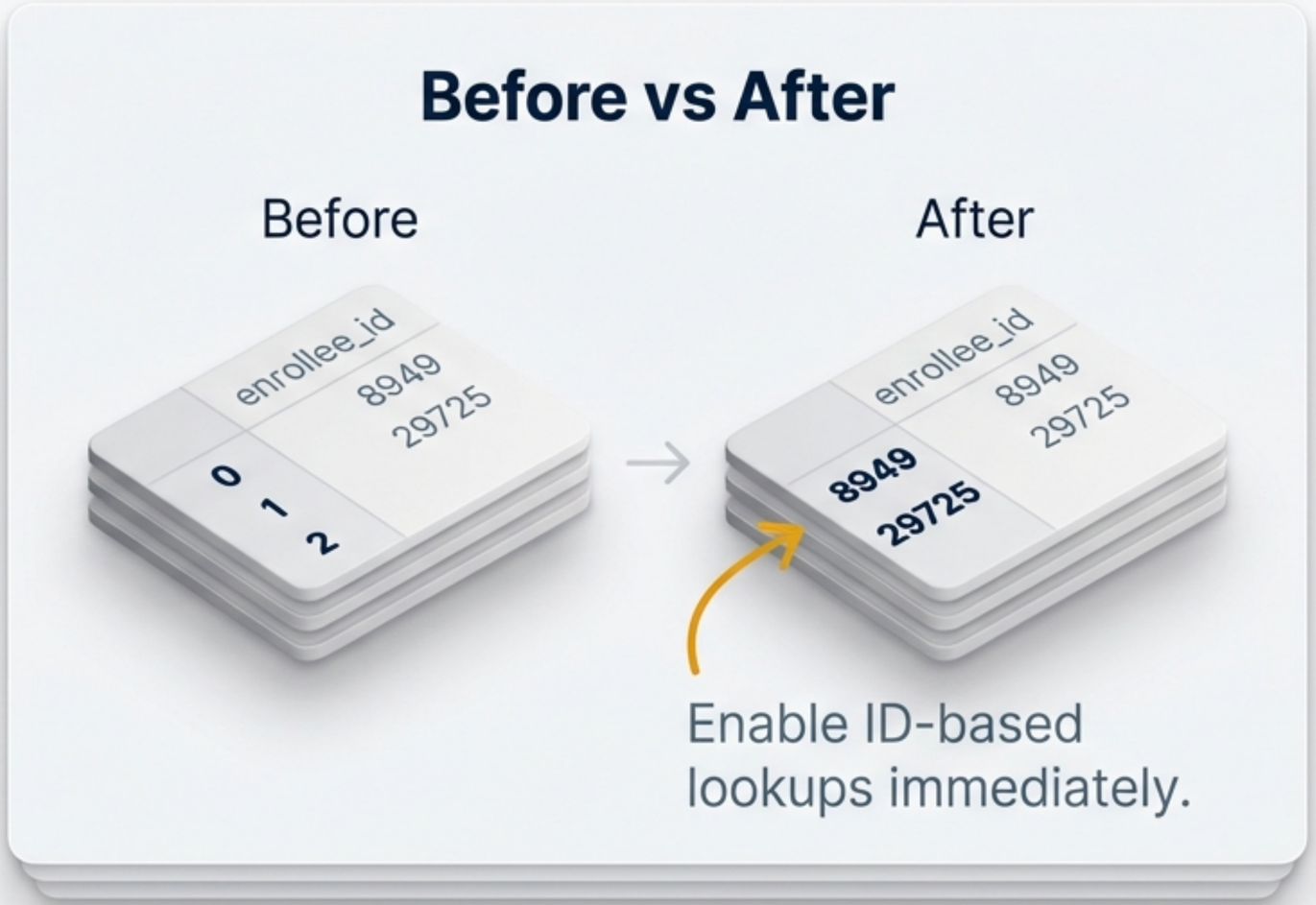
```
pd.read_csv(..., header=1)
```

	Generated by System X, Date: 2023-10-27	
1	**ID**	**Name**
2	101	Alice
4	103	Alice
5	104	Erica
6	105	Romana

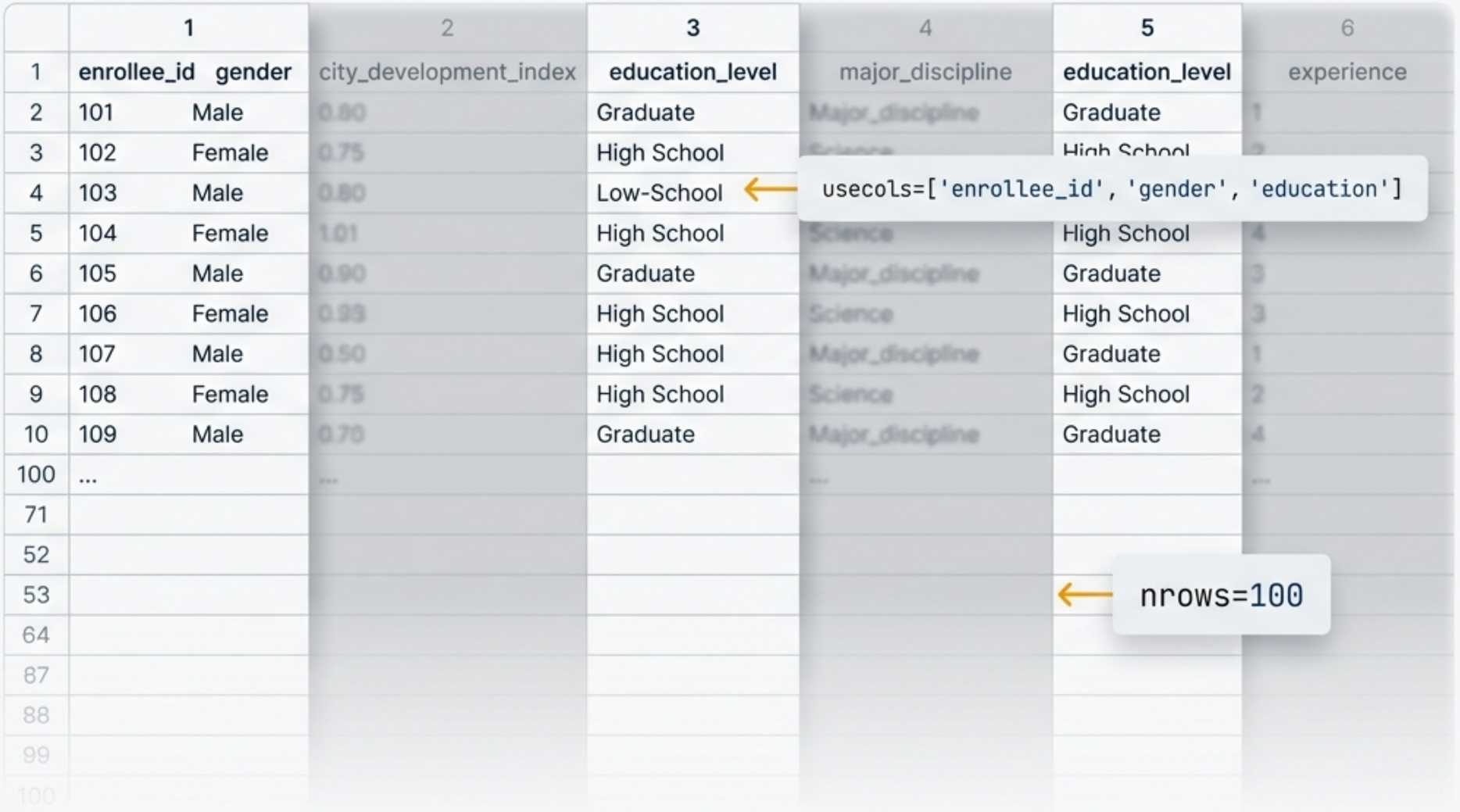
Header is not in the first row. The `header` argument specifies the correct row index.

Removing Redundant Indexing

```
pd.read_csv('aug_train.csv',
    index_col='enrollee_id')
```



Strategic Loading: Selecting Only What You Need



	1		2	3	4	5	6
1	enrollee_id	gender	city_development_index	education_level	major_discipline	education_level	experience
2	101	Male	0.80	Graduate	Major_discipline	Graduate	1
3	102	Female	0.75	High School	Science	High School	2
4	103	Male	0.80	Low-School	Major_discipline	Graduate	1
5	104	Female	1.01	High School	Science	High School	4
6	105	Male	0.90	Graduate	Major_discipline	Graduate	3
7	106	Female	0.93	High School	Science	High School	3
8	107	Male	0.50	High School	Major_discipline	Graduate	1
9	108	Female	0.75	High School	Science	High School	2
10	109	Male	0.70	Graduate	Major_discipline	Graduate	4
100	...		...		...		...
71							
52							
53							
64							
87							
88							
99							
100							

Pro Tip:

usecols drastically reduces memory usage on import.

Handling ‘Alien’ Characters

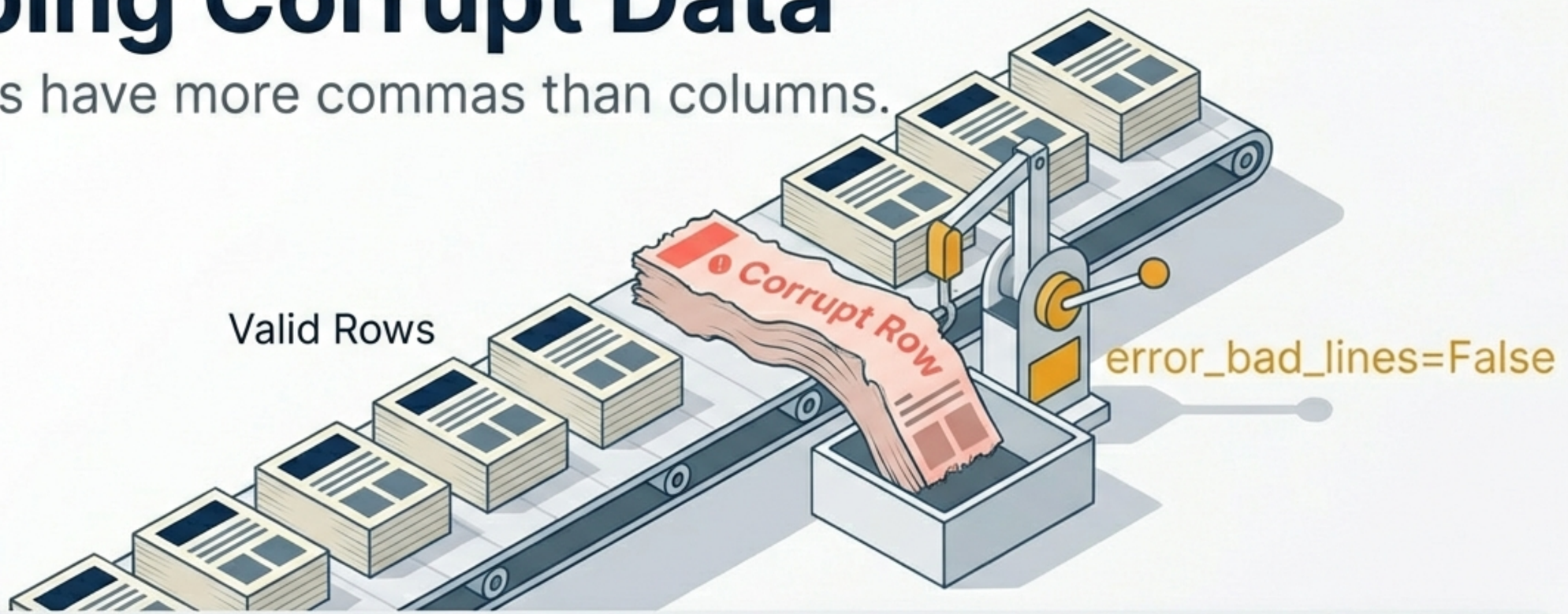
Solving UnicodeDecodeError

Also try 'cp1252' or inspect file in a text editor.

```
pd.read_csv('zomato.csv', encoding='latin-1')
```

Skipping Corrupt Data

When rows have more commas than columns.



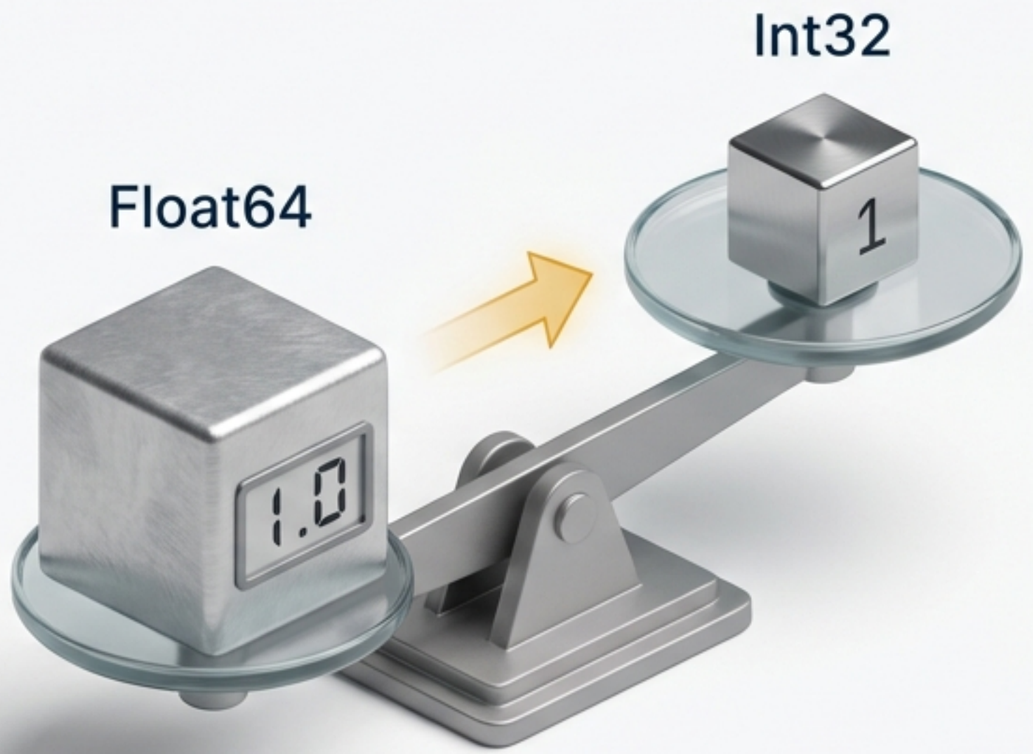

```
pd.read_csv('books.csv', error_bad_lines=False)
```

Prevents parser crashes by discarding malformed rows.

Optimising Memory with Data Types

Converting types at source.

```
pd.read_csv('aug_train.csv',
    dtype={'target': int})
```



Convert binary 1.0/0.0 targets to Integers to save RAM.

Mastering Time Series Data

From String Objects to Datetime.

2008-04-18

Object (String)

Datetime64

```
pd.read_csv('IPL_matches.csv', parse_dates=['date'])
```

Pro Tip

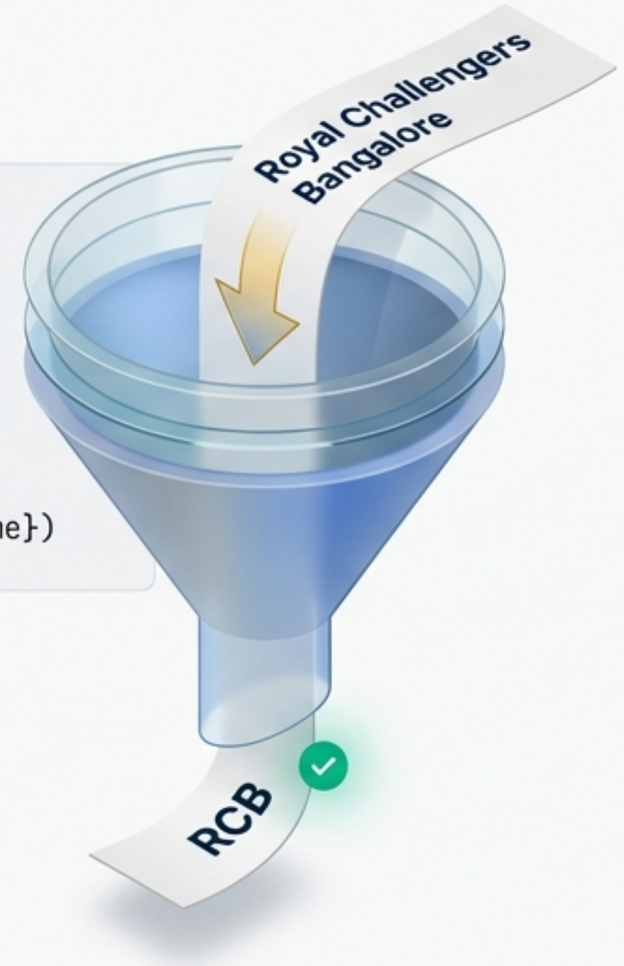
To merge separate columns (Year, Month, Day) into one, pass a list of lists:

parse_dates=[['year', 'month', 'day']]

Custom Logic and Data Hygiene

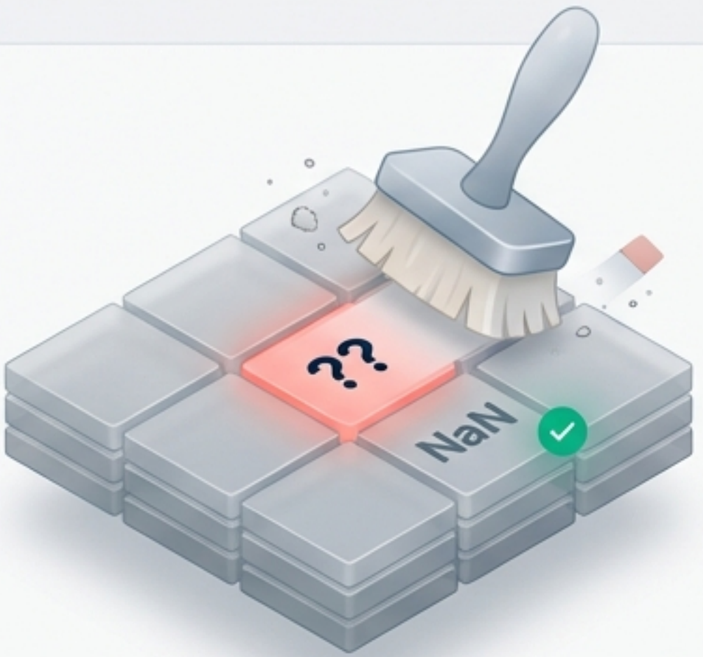
Converters (Transform on Load)

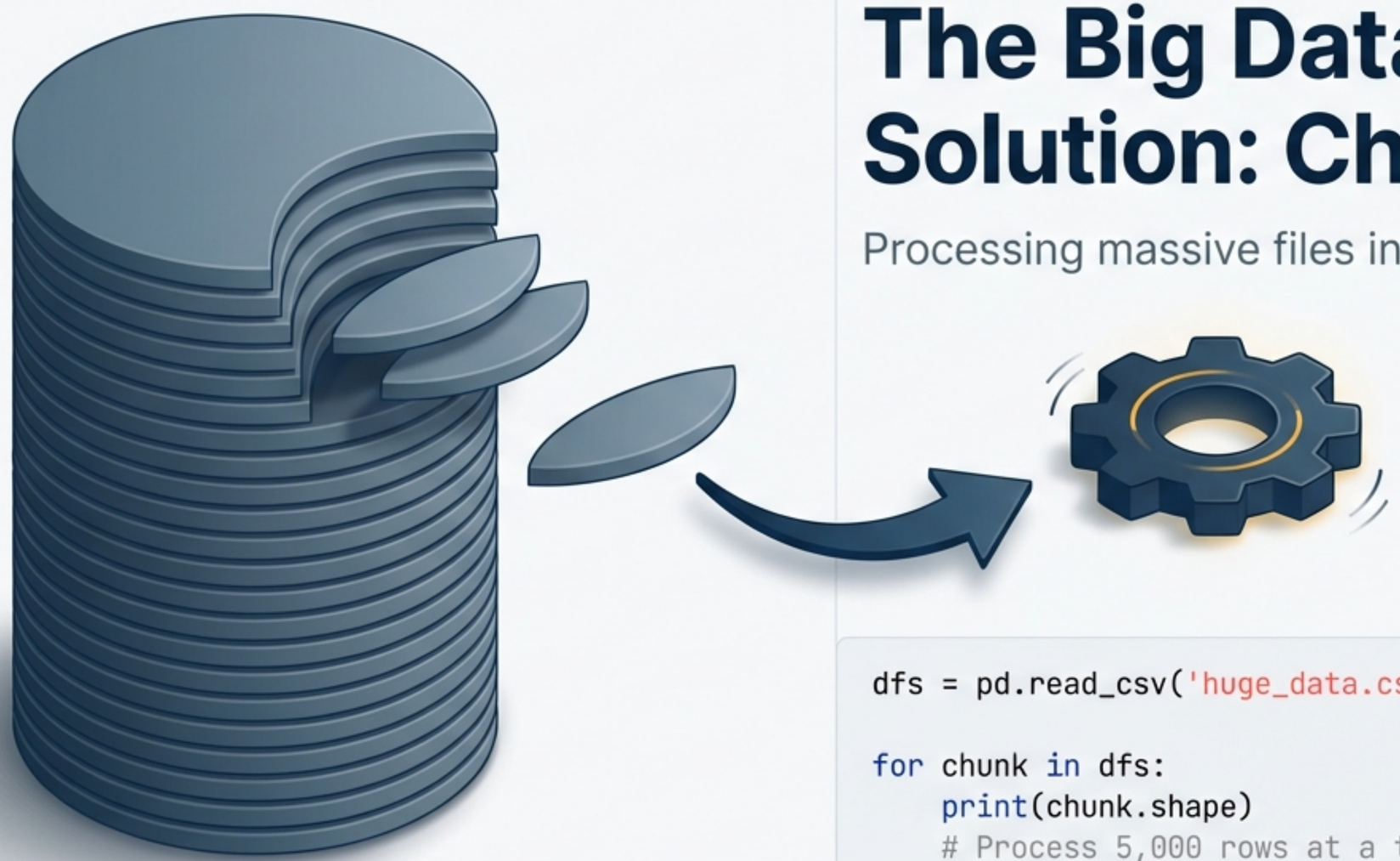
```
def rename(name):
    if name ==
        name== "Royal Challengers Bangalore":
        return "RCB"
    return name


pd.read_csv('ipl.csv', converters={'team1': rename})
```



Handling Custom Missing Values

```
na_values=['-', '??']
```





The Big Data Solution: Chunking

Processing massive files in batches.

```
dfs = pd.read_csv('huge_data.csv', chunksize=5000)

for chunk in dfs:
    print(chunk.shape)
    # Process 5,000 rows at a time
```

Avoid Memory Errors by processing in loops rather than loading all at once.

Your read_csv Arsenal

Structure	Filtering	Hygiene	Transform	Scale
`sep`	`usecols`	`encoding`	`dtype`	`chunksize`
`delimiter`	`squeeze`	`error_bad_lines`	`parse_dates`	
`header`	`skiprows`	`na_values`	`converters`	
`names`	`nrows`			
`index_col`				

Mastering these 15 parameters covers 90% of data engineering challenges.