Regex, the 20 pieces that cover 99%.

D8A.ACADEMY/RESOURCES/REGEX-FOR-DATA

ANCHORS & CLASSES

`^start · end$`	edges of the string
`\d \w \s`	digit word space
`.`	any single char
`[A-Za-z]`	build your own class

QUANTIFIERS

`+ · *`	1+ · 0+ repeats
`?`	optional
`{2,4}`	between 2 and 4
`.*`	← greedy. eats everything.

GROUPS

`(abc)`	capture it
`(?:abc)`	group, no capture
`a|b`	or
`\1`	reuse capture 1

PATTERNS TO STEAL

`\d{4}-\d{2}-\d{2}`	ISO date
`[\w.]+@[\w.]+\.\w+`	email-ish
`\d+[.,]\d+`	decimal number
`\s{2,}`	extra spaces

IN PANDAS

`s.str.contains(r'…')`	filter rows
`s.str.extract(r'(…)')`	pull the group
`s.str.replace(r'…','')`	clean

IN SQL

`x ~ 'regex'`	postgres match
`REGEXP_LIKE(x, r)`	most other dialects
`REGEXP_REPLACE(x,r,'')`	clean in place

WORTH MEMORIZING

01 Columns out of chaos

One messy log column in, three clean columns out.

```
# "order #4521 shipped 2026-08-12 (US)"
s = df["raw"]
df["order_id"] = s.str.extract(r"#(\d+)")
df["date"] = s.str.extract(
    r"(\d{4}-\d{2}-\d{2})")
df["country"] = s.str.extract(
    r"\((\w{2})\)")
```

→ extract returns whatever the (group) captures. One pattern per concept beats one giant pattern.

02 Trap: greedy by default

The star eats as much as it can, then walks back. Usually too far.

PATTERN	ON THIS INPUT	IT MATCHES
`<b>(.*)</b>`	`<b>a</b> <b>b</b>`	`a</b> <b>b : too much`
`<b>(.*?)</b>`	`<b>a</b> <b>b</b>`	`a : lazy stops early`
`^\d+$`	`abc123`	`nothing: anchors save you`

→ Add ? after * or + to make it lazy. And anchor with ^ $ whenever you mean the whole string.