FOUNDATION LESSON

WEEK 4 · LESSON 12

# Build a validated DataFrame report

Clean text deliberately, validate records and reconcile the final report

THE IDEA BEHIND THE OPERATION

MENTAL MODEL

# The idea behind the operation

Clean text deliberately, validate records and reconcile the final report

# NORMALISE TEXT WITHOUT INVENTING MISSING CODES

CODE WALKTHROUGH W04-L12-S03

```
01  import pandas as pd
02  codes = pd.Series([" a ", None, "b"], dtype="string")
03  clean = codes.str.strip().str.upper()
04  print(clean.fillna("MISSING").tolist())
05  print("Missing:", int(clean.isna().sum()))
```

## WHAT TO TRACK

Codes are uppercase; absent codes remain absent.

## PREDICT THE RESULT

Predict the output.
Explain each operation.
Pause before continuing.

# NORMALISE TEXT WITHOUT INVENTING MISSING CODES — RESULT

CODE WALKTHROUGH W04-L12-S04

```
01  import pandas as pd
02  codes = pd.Series([" a ", None, "b"], dtype="string")
03  clean = codes.str.strip().str.upper()
04  print(clean.fillna("MISSING").tolist())
05  print("Missing:", int(clean.isna().sum()))
```

## WHAT TO TRACK

Codes are uppercase; absent codes remain absent.

## VALIDATE THE RESULT

```
['A', 'MISSING', 'B']
Missing: 1
```

# VALIDATE AND RECONCILE THE SUPPLIED REPORT

CODE WALKTHROUGH W04-L12-S05

```
01  import pandas as pd
02  df = pd.read_csv("data/sales.csv", encoding="utf-8", dtype={"pr
↪   oduct":"string"})
03  df["product"] = df["product"].str.strip().str.upper()
04  df["date"] = pd.to_datetime(df["date"], format="%Y-%m-%d", erro
↪   rs="coerce")
05  valid = (df["product"].notna() & df["product"].ne("")
06           & df["date"].notna() & df["units"].ge(0) & df["price"]
↪   .ge(0))
07  accepted = df.loc[valid].copy()
08  accepted["revenue"] = accepted["units"] * accepted["price"]
09  print("Accepted:", len(accepted), "Rejected:", int((~valid).sum
↪   ()))
10  print(accepted[["product","revenue"]].to_string(index=False))
11  print("Total EUR:", accepted["revenue"].sum())
```

## WHAT TO TRACK

Missing/empty codes, invalid dates and negative values are rejected.

## PREDICT THE RESULT

Predict the output.
Explain each operation.
Pause before continuing.

## VALIDATE AND RECONCILE THE SUPPLIED REPORT — RESULT

CODE WALKTHROUGH W04-L12-S06

```
01  import pandas as pd
02  df = pd.read_csv("data/sales.csv", encoding="utf-8", dtype={"pr
↪   oduct":"string"})
03  df["product"] = df["product"].str.strip().str.upper()
04  df["date"] = pd.to_datetime(df["date"], format="%Y-%m-%d", erro
↪   rs="coerce")
05  valid = (df["product"].notna() & df["product"].ne("")
06           & df["date"].notna() & df["units"].ge(0) & df["price"]
↪   .ge(0))
07  accepted = df.loc[valid].copy()
08  accepted["revenue"] = accepted["units"] * accepted["price"]
09  print("Accepted:", len(accepted), "Rejected:", int((~valid).sum
↪   ()))
10  print(accepted[["product","revenue"]].to_string(index=False))
11  print("Total EUR:", accepted["revenue"].sum())
```

### WHAT TO TRACK

Missing/empty codes, invalid dates and negative values are rejected.

### VALIDATE THE RESULT

```
Accepted: 3 Rejected: 1
product  revenue
      A     24.0
      B     15.0
      D      0.0
Total EUR: 39.0
```

# ALL RECORDS CAN BE REJECTED

CODE WALKTHROUGH W04-L12-S07

```
01  import pandas as pd
02  df = pd.DataFrame({"product":["",None], "units":[-1,2]})
03  df["product"] = df["product"].astype("string")
04  valid = (df["product"].notna() & df["product"].ne("")
05           & df["units"].ge(0)).fillna(False)
06  accepted = df.loc[valid]
07  print("Accepted:", len(accepted))
08  if accepted.empty:
09      raise ValueError("no valid records for this report")
```

WHAT TO TRACK

A completed filter is not proof that useful data remains.

PREDICT THE RESULT

Predict the output.
Explain each operation.
Pause before continuing.

## ALL RECORDS CAN BE REJECTED — RESULT

CODE WALKTHROUGH W04-L12-S08

```
01  import pandas as pd
02  df = pd.DataFrame({"product":["",None], "units":[-1,2]})
03  df["product"] = df["product"].astype("string")
04  valid = (df["product"].notna() & df["product"].ne("")
05           & df["units"].ge(0)).fillna(False)
06  accepted = df.loc[valid]
07  print("Accepted:", len(accepted))
08  if accepted.empty:
09      raise ValueError("no valid records for this report")
```

### WHAT TO TRACK

A completed filter is not proof that useful data remains.

### VALIDATE THE RESULT

```
Accepted: 0
ValueError: no valid records
for this report
```

GUIDED MODIFICATION

LEARNER PRACTICE

# Guided modification

Normalise product codes [ a , B, missing] using string dtype, strip and upper. Report one missing code. Filter to present non-empty codes without replacing the missing code with a product.

PREDICT · MODIFY · RUN

01 Restate the supplied rules.

02 Predict the result.

03 Modify and run the code.

04 Compare actual output.

05 Explain any difference.

INDEPENDENT APPLICATION

LEARNER PRACTICE

# Independent application

Rebuild the sales.csv report, preserve raw date text, export accepted_report.csv and rejected_report.csv, and verify accepted plus rejected equals input rows. State the accepted revenue total and test the all-rejected boundary.

PREDICT · MODIFY · RUN

01 Restate the supplied rules.

02 Predict the result.

03 Modify and run the code.

04 Compare actual output.

05 Explain any difference.

RECAP AND NEXT STEP

# Recap and next step

You can now create and inspect numerical arrays, reason about selection, broadcasting and reductions, and build labelled DataFrame reports. Keep units, labels and validation policies explicit. Complete the required weekly quiz and ninety-minute project. Lesson practice remains formative.

UNDERSTAND APPLY VALIDATE EXPLAIN