FOUNDATION LESSON

WEEK 5 · LESSON 1

Missing data and deliberate policies

Missing means unknown until a stated business rule justifies a replacement.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Missing means unknown until a stated business rule justifies a replacement.

COUNT MISSING VALUES

CODE WALKTHROUGH W05-L01-S03

```
01  import pandas as pd
02  sales = pd.Series([100.0, None, 0.0])
03  print(sales.isna().tolist())
04  print("Missing:", sales.isna().sum())
05  print("Observed mean:", sales.mean())
```

WHAT TO TRACK

Count missing values

PREDICT THE RESULT

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

COUNT MISSING VALUES · RESULT

CODE WALKTHROUGH

W05-L01-S04

```
01  import pandas as pd
02  sales = pd.Series([100.0, None, 0.0])
03  print(sales.isna().tolist())
04  print("Missing:", sales.isna().sum())
05  print("Observed mean:", sales.mean())
```

WHAT TO TRACK

Count missing values

VALIDATE THE RESULT

```
[False, True, False]
Missing: 1
Observed mean: 50.0
```

A REQUIRED INVOICE AMOUNT

CODE WALKTHROUGH W05-L01-S05

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "B", "C"],
03                     "amount": [100, None, 0]})
04  accepted = df.loc[df["amount"].notna()].copy()
05  rejected = df.loc[df["amount"].isna()].copy()
06  print("Accepted:", accepted["id"].tolist())
07  print("Rejected:", rejected["id"].tolist())
08  print("Total:", accepted["amount"].sum())
```

WHAT TO TRACK

A required invoice amount

PREDICT THE RESULT

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

A REQUIRED INVOICE AMOUNT · RESULT

CODE WALKTHROUGH W05-L01-S06

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "B", "C"],
03                     "amount": [100, None, 0]})
04  accepted = df.loc[df["amount"].notna()].copy()
05  rejected = df.loc[df["amount"].isna()].copy()
06  print("Accepted:", accepted["id"].tolist())
07  print("Rejected:", rejected["id"].tolist())
08  print("Total:", accepted["amount"].sum())
```

WHAT TO TRACK

A required invoice amount

VALIDATE THE RESULT

```
Accepted: ['A', 'C']
Rejected: ['B']
Total: 100.0
```

ALL VALUES ARE ABSENT

CODE WALKTHROUGH W05-L01-S07

```
01  import pandas as pd
02  values = pd.Series([None, None], dtype="float64")
03  print("Observed:", values.count())
04  print("Guarded sum:", values.sum(min_count=1))
```

WHAT TO TRACK

All values are absent

PREDICT THE RESULT

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

ALL VALUES ARE ABSENT · RESULT

CODE WALKTHROUGH W05-L01-S08

```
01  import pandas as pd
02  values = pd.Series([None, None], dtype="float64")
03  print("Observed:", values.count())
04  print("Guarded sum:", values.sum(min_count=1))
```

WHAT TO TRACK

All values are absent

VALIDATE THE RESULT

```
Observed: 0
Guarded sum: nan
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Use [50, None, 0, 70]. Print missing count and observed mean. Keep the known zero.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Create amounts [None, None]. Print a total only if at least one amount is observed; otherwise print No observations.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does fillna(0) merely change the display, or does it assert that every unknown value is a known zero? It changes the analytical assumption.

RECAP AND NEXT STEP

Recap and next step

Count missing values before changing them. Preserve known zeroes, document exclusions, and guard totals when nothing was observed.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 2

Duplicates and safe numeric conversion

A duplicate is defined by a business key and a revision rule.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A duplicate is defined by a business key and a revision rule.

EXACT DUPLICATES

CODE WALKTHROUGH W05-L02-S03

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "A", "B"],
03                     "amount": [10, 10, 20]})
04  print(df.duplicated().tolist())
05  print(len(df.drop_duplicates()))
```

WHAT TO TRACK

Exact duplicates

PREDICT THE RESULT

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

EXACT DUPLICATES · RESULT

CODE WALKTHROUGH W05-L02-S04

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "A", "B"],
03                     "amount": [10, 10, 20]})
04  print(df.duplicated().tolist())
05  print(len(df.drop_duplicates()))
```

WHAT TO TRACK

Exact duplicates

VALIDATE THE RESULT

```
[False, True, False]
2
```

LATEST REVISION BEFORE VALIDATION

CODE WALKTHROUGH W05-L02-S05

```
01 import pandas as pd
02 df = pd.DataFrame({"id": ["A", "A", "B"],
03  "revision": [1, 2, 1], "amount": ["10", "12", "oops"]})
04 latest = df.sort_values("revision").drop_duplicates(
05     "id", keep="last").copy()
06 latest["amount"] = pd.to_numeric(
07     latest["amount"], errors="coerce")
08 print(latest.sort_values("id").to_string(index=False))
09 print("Invalid:", latest["amount"].isna().sum())
```

WHAT TO TRACK

Latest revision before validation

PREDICT THE RESULT

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

LATEST REVISION BEFORE VALIDATION · RESULT

CODE WALKTHROUGH W05-L02-S06

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "A", "B"],
03   "revision": [1, 2, 1], "amount": ["10", "12", "oops"]})
04  latest = df.sort_values("revision").drop_duplicates(
05      "id", keep="last").copy()
06  latest["amount"] = pd.to_numeric(
07      latest["amount"], errors="coerce")
08  print(latest.sort_values("id").to_string(index=False))
09  print("Invalid:", latest["amount"].isna().sum())
```

WHAT TO TRACK

Latest revision before validation

VALIDATE THE RESULT

```
id  revision  amount
 A         2    12.0
 B         1     NaN
Invalid: 1
```

CONFLICTING SAME-REVISION RECORDS

CODE WALKTHROUGH W05-L02-S07

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "A"],
03   "revision": [2, 2], "amount": [12, 15]})
04  if df.duplicated(["id", "revision"]).any():
05      raise ValueError("Resolve repeated invoice revision")
```

WHAT TO TRACK

Conflicting same-revision records

PREDICT THE RESULT

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

CONFLICTING SAME-REVISION RECORDS · RESULT

CODE WALKTHROUGH W05-L02-S08

```
01  import pandas as pd
02  df = pd.DataFrame({"id": ["A", "A"],
03   "revision": [2, 2], "amount": [12, 15]})
04  if df.duplicated(["id", "revision"]).any():
05      raise ValueError("Resolve repeated invoice revision")
```

WHAT TO TRACK

Conflicting same-revision records

VALIDATE THE RESULT

ValueError: Resolve repeated invoice revision

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Convert ["5", "bad", "0"] to numbers.
Print the invalid count and valid total.

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 PRACTICE

LEARNER PRACTICE

Independent practice

For invoice X revisions 1 and 3 with amounts 9 and 11, select revision 3 and print 11.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: If the latest revision has invalid text, may we silently restore an older valid amount? No. That would introduce an unstated revision policy.

RECAP AND NEXT STEP

Recap and next step

Define identity and revision priority before deduplicating. Record conversion failures and validate business rules separately.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 3

Tidy column names and groupby

One row has a stated grain; grouping partitions those rows by a key.

THE MENTAL MODEL

MENTAL MODEL

The mental model

One row has a stated grain; grouping partitions those rows by a key.

RENAME A COLUMN

CODE WALKTHROUGH W05-L03-S03

```
01  import pandas as pd
02  df = pd.DataFrame({"Sales EUR": [10, 20]})
03  clean = df.rename(columns={"Sales EUR": "sales_eur"})
04  print(clean.columns.tolist())
05  print(clean["sales_eur"].sum())
```

WHAT TO TRACK

Rename a column

PREDICT THE RESULT

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

RENAME A COLUMN · RESULT

CODE WALKTHROUGH W05-L03-S04

```
01  import pandas as pd
02  df = pd.DataFrame({"Sales EUR": [10, 20]})
03  clean = df.rename(columns={"Sales EUR": "sales_eur"})
04  print(clean.columns.tolist())
05  print(clean["sales_eur"].sum())
```

WHAT TO TRACK

Rename a column

VALIDATE THE RESULT

```
['sales_eur']
30
```

REVENUE BY REGION

CODE WALKTHROUGH W05-L03-S05

```
01  import pandas as pd
02  df = pd.DataFrame({"region": ["East", "West", "East"],
03                     "revenue_eur": [100, 80, 50]})
04  summary = df.groupby("region")["revenue_eur"].sum()
05  print(summary.to_string())
06  print("Control:", summary.sum(), df["revenue_eur"].sum())
```

WHAT TO TRACK

Revenue by region

PREDICT THE RESULT

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

REVENUE BY REGION · RESULT

CODE WALKTHROUGH W05-L03-S06

```
01  import pandas as pd
02  df = pd.DataFrame({"region": ["East", "West", "East"],
03                     "revenue_eur": [100, 80, 50]})
04  summary = df.groupby("region")["revenue_eur"].sum()
05  print(summary.to_string())
06  print("Control:", summary.sum(), df["revenue_eur"].sum())
```

WHAT TO TRACK

Revenue by region

VALIDATE THE RESULT

```
region
East    150
West     80
Control: 230 230
```

AN UNKNOWN REGION

CODE WALKTHROUGH W05-L03-S07

```
01  import pandas as pd
02  df = pd.DataFrame({"region": ["East", None],
03                     "amount": [10, 20]})
04  print(df.groupby("region", dropna=False)["amount"].sum())
```

WHAT TO TRACK

An unknown region

PREDICT THE RESULT

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

AN UNKNOWN REGION · RESULT

CODE WALKTHROUGH W05-L03-S08

```
01  import pandas as pd
02  df = pd.DataFrame({"region": ["East", None],
03                     "amount": [10, 20]})
04  print(df.groupby("region", dropna=False)["amount"].sum())
```

WHAT TO TRACK

An unknown region

VALIDATE THE RESULT

```
region
East    10
NaN     20
Name: amount, dtype: int64
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Group A, B, A with amounts 5, 7, 8. Print group totals and the overall total.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Rename Amount EUR to amount_eur and group by region, retaining a missing region.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does a regional sum retain one row per order? No. It reduces the table to one summary per region.

RECAP AND NEXT STEP

Recap and next step

Name variables clearly, state the row grain, choose the grouping key, and reconcile the result.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 4

Multiple aggregations and transform

Aggregation reduces groups; transform returns a value aligned with each input row.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Aggregation reduces groups; transform returns a value aligned with each input row.

SEVERAL STATISTICS TOGETHER

CODE WALKTHROUGH W05-L04-S03

```
01  import pandas as pd
02  d=pd.DataFrame({"team":["A","A","B"],"sales":[10,30,20]})
03  s=d.groupby("team").agg(total=("sales","sum"),
04                          average=("sales","mean"),
05                          rows=("sales","size"))
06  print(s.to_string())
```

WHAT TO TRACK

Several statistics together

PREDICT THE RESULT

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

SEVERAL STATISTICS TOGETHER · RESULT

CODE WALKTHROUGH W05-L04-S04

```
01  import pandas as pd
02  d=pd.DataFrame({"team":["A","A","B"],"sales":[10,30,20]})
03  s=d.groupby("team").agg(total=("sales","sum"),
04                          average=("sales","mean"),
05                          rows=("sales","size"))
06  print(s.to_string())
```

WHAT TO TRACK

Several statistics together

VALIDATE THE RESULT

```
      total  average  rows
team
A        40     20.0     2
B        20     20.0     1
```

ORDER SHARES WITHIN A REGION

CODE WALKTHROUGH W05-L04-S05

```
01  import pandas as pd
02  d=pd.DataFrame({"region":["N","N","S"],"sales":[10,30,20]})
03  totals=d.groupby("region")["sales"].transform("sum")
04  d["share"]=d["sales"]/totals.where(totals>0)
05  print(d.to_string(index=False))
```

WHAT TO TRACK

Order shares within a region

PREDICT THE RESULT

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

ORDER SHARES WITHIN A REGION · RESULT

CODE WALKTHROUGH W05-L04-S06

```
01  import pandas as pd
02  d=pd.DataFrame({"region":["N","N","S"],"sales":[10,30,20]})
03  totals=d.groupby("region")["sales"].transform("sum")
04  d["share"]=d["sales"]/totals.where(totals>0)
05  print(d.to_string(index=False))
```

WHAT TO TRACK

Order shares within a region

VALIDATE THE RESULT

```
region  sales  share
     N     10   0.25
     N     30   0.75
     S     20   1.00
```

A ZERO-TOTAL GROUP

CODE WALKTHROUGH W05-L04-S07

```
01  import pandas as pd
02  d=pd.DataFrame({"group":["A","A"],"sales":[0,0]})
03  t=d.groupby("group")["sales"].transform("sum")
04  print((d["sales"]/t.where(t>0)).tolist())
```

WHAT TO TRACK

A zero-total group

PREDICT THE RESULT

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

A ZERO-TOTAL GROUP · RESULT

CODE WALKTHROUGH W05-L04-S08

```
01  import pandas as pd
02  d=pd.DataFrame({"group":["A","A"],"sales":[0,0]})
03  t=d.groupby("group")["sales"].transform("sum")
04  print((d["sales"]/t.where(t>0)).tolist())
```

WHAT TO TRACK

A zero-total group

VALIDATE THE RESULT

[nan, nan]

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

For group A values 2 and 6, attach each row’s group mean with transform.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Compare size and count for amounts [5, None] in group A.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does transform remove repeated group labels? No. Its result remains aligned with the input rows.

RECAP AND NEXT STEP

Recap and next step

Use agg for one summary per group and transform for aligned per-row statistics.
Guard denominators and name counts precisely.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 5

Merging and stacking tables safely

A merge matches keys; concat stacks compatible rows.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A merge matches keys; concat stacks compatible rows.

A MANY-TO-ONE LOOKUP

CODE WALKTHROUGH W05-L05-S03

```
01  import pandas as pd
02  orders=pd.DataFrame({"customer":["A","A","B"],"amount":[10,20,3
↪   0]})
03  customers=pd.DataFrame({"customer":["A","B"],"region":["N","S"]
↪   })
04  joined=orders.merge(customers,on="customer",how="left",
05                      validate="many_to_one",indicator=True)
06  print(joined.to_string(index=False))
07  print("Total:", joined["amount"].sum())
```

WHAT TO TRACK

A many-to-one lookup

PREDICT THE RESULT

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

A MANY-TO-ONE LOOKUP · RESULT

CODE WALKTHROUGH W05-L05-S04

```
01  import pandas as pd
02  orders=pd.DataFrame({"customer":["A","A","B"],"amount":[10,20,3
↪   0]})
03  customers=pd.DataFrame({"customer":["A","B"],"region":["N","S"]
↪   })
04  joined=orders.merge(customers,on="customer",how="left",
05                      validate="many_to_one",indicator=True)
06  print(joined.to_string(index=False))
07  print("Total:", joined["amount"].sum())
```

WHAT TO TRACK

A many-to-one lookup

VALIDATE THE RESULT

```
customer  amount region
_merge
       A      10      N
both
       A      20      N
both
       B      30      S
both
Total: 60
```

STACK TWO DELIVERY BATCHES

CODE WALKTHROUGH W05-L05-S05

```
01  import pandas as pd
02  a=pd.DataFrame({"id":["X"],"amount":[10]})
03  b=pd.DataFrame({"id":["Y"],"amount":[20]})
04  both=pd.concat([a,b],ignore_index=True)
05  print(both.to_string(index=False))
06  print("Rows:", len(both))
```

WHAT TO TRACK

Stack two delivery batches

PREDICT THE RESULT

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

STACK TWO DELIVERY BATCHES · RESULT

CODE WALKTHROUGH W05-L05-S06

```
01  import pandas as pd
02  a=pd.DataFrame({"id":["X"],"amount":[10]})
03  b=pd.DataFrame({"id":["Y"],"amount":[20]})
04  both=pd.concat([a,b],ignore_index=True)
05  print(both.to_string(index=False))
06  print("Rows:", len(both))
```

WHAT TO TRACK

Stack two delivery batches

VALIDATE THE RESULT

```
id  amount
 X      10
 Y      20
Rows: 2
```

A DUPLICATED LOOKUP KEY

CODE WALKTHROUGH W05-L05-S07

```
01  import pandas as pd
02  o=pd.DataFrame({"customer":["A"],"amount":[10]})
03  c=pd.DataFrame({"customer":["A","A"],"region":["N","S"]})
04  o.merge(c,on="customer",validate="many_to_one")
```

WHAT TO TRACK

A duplicated lookup key

PREDICT THE RESULT

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

A DUPLICATED LOOKUP KEY · RESULT

CODE WALKTHROUGH W05-L05-S08

```
01 import pandas as pd
02 o=pd.DataFrame({"customer":["A"],"amount":[10]})
03 c=pd.DataFrame({"customer":["A","A"],"region":["N","S"]})
04 o.merge(c,on="customer",validate="many_to_one")
```

WHAT TO TRACK

A duplicated lookup key

VALIDATE THE RESULT

```
pandas.errors.MergeError:
Merge keys are not unique in
right dataset; not a
many-to-one merge
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Left-merge orders for A and Z with a lookup containing only A. Use indicator and print the unmatched customer.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Concatenate two one-row batches with the same invoice id, then detect the repeated id.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does a successful left merge guarantee every key matched? No. Inspect the merge indicator.

RECAP AND NEXT STEP

Recap and next step

Declare join cardinality, inspect unmatched rows, and reconcile totals. Use concat to stack batches with compatible schemas.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 6

Pivot tables and integrated cleaning

A pivot summary changes the display grain and needs an explicit aggregation rule.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A pivot summary changes the display grain and needs an explicit aggregation rule.

REGION BY MONTH

CODE WALKTHROUGH W05-L06-S03

```
01  import pandas as pd
02  d=pd.DataFrame({"region":["N","N","S"],
03   "month":["Jan","Feb","Jan"],"amount":[10,20,30]})
04  p=d.pivot_table(index="region",columns="month",values="amount",
05                  aggfunc="sum",fill_value=0)
06  print(p.to_string())
07  print("Control:", p.to_numpy().sum())
```

WHAT TO TRACK

Region by month

PREDICT THE RESULT

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

REGION BY MONTH · RESULT

CODE WALKTHROUGH W05-L06-S04

```
01  import pandas as pd
02  d=pd.DataFrame({"region":["N","N","S"],
03   "month":["Jan","Feb","Jan"],"amount":[10,20,30]})
04  p=d.pivot_table(index="region",columns="month",values="amount",
05                  aggfunc="sum",fill_value=0)
06  print(p.to_string())
07  print("Control:", p.to_numpy().sum())
```

WHAT TO TRACK

Region by month

VALIDATE THE RESULT

```
month   Feb  Jan
region
N        20   10
S         0   30
Control: 60
```

CLEAN, JOIN, SUMMARISE

CODE WALKTHROUGH W05-L06-S05

```
01  import pandas as pd
02  o=pd.DataFrame({"customer":["A","B","A"],
03   "amount":["10","bad","20"]})
04  c=pd.DataFrame({"customer":["A","B"],"region":["N","S"]})
05  o["amount"]=pd.to_numeric(o["amount"],errors="coerce")
06  valid=o.loc[o["amount"].notna()].copy()
07  j=valid.merge(c,on="customer",how="left",validate="many_to_one"
↪   )
08  print("Rejected:", len(o)-len(valid))
09  print(j.groupby("region")["amount"].sum().to_dict())
```

WHAT TO TRACK

Clean, join, summarise

PREDICT THE RESULT

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

CLEAN, JOIN, SUMMARISE · RESULT

CODE WALKTHROUGH W05-L06-S06

```
01 import pandas as pd
02 o=pd.DataFrame({"customer":["A","B","A"],
03  "amount":["10","bad","20"]})
04 c=pd.DataFrame({"customer":["A","B"],"region":["N","S"]})
05 o["amount"]=pd.to_numeric(o["amount"],errors="coerce")
06 valid=o.loc[o["amount"].notna()].copy()
07 j=valid.merge(c,on="customer",how="left",validate="many_to_one"
↪ )
08 print("Rejected:", len(o)-len(valid))
09 print(j.groupby("region")["amount"].sum().to_dict())
```

WHAT TO TRACK

Clean, join, summarise

VALIDATE THE RESULT

```
Rejected: 1
{'N': 30.0}
```

AN INVALID PIVOT GRAIN

CODE WALKTHROUGH W05-L06-S07

```
01  import pandas as pd
02  d=pd.DataFrame({"region":["N","N"],
03   "month":["Jan","Jan"],"amount":[10,20]})
04  print(d.pivot(index="region",columns="month",values="amount"))
```

WHAT TO TRACK

An invalid pivot grain

PREDICT THE RESULT

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

AN INVALID PIVOT GRAIN · RESULT

CODE WALKTHROUGH W05-L06-S08

```
01  import pandas as pd
02  d=pd.DataFrame({"region":["N","N"],
03   "month":["Jan","Jan"],"amount":[10,20]})
04  print(d.pivot(index="region",columns="month",values="amount"))
```

WHAT TO TRACK

An invalid pivot grain

VALIDATE THE RESULT

```
ValueError: Index contains
duplicate entries, cannot
reshape
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Pivot N/Jan amounts 4 and 6 using sum.
Check that the single cell is 10.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Clean ["8", "bad", "-2"] under the rule numeric and nonnegative. Print accepted total and rejected count.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: May fill_value=0 prove no orders occurred? No. That meaning must come from the dataset’s coverage rules.

RECAP AND NEXT STEP

Recap and next step

Validate before reshaping. Declare the pivot aggregation and missing-cell meaning, then reconcile the accepted total.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 7

Analytical questions and Matplotlib objects

The Figure is the whole canvas; an Axes is a plotting area on that canvas.

THE MENTAL MODEL

MENTAL MODEL

The mental model

The Figure is the whole canvas; an Axes is a plotting area on that canvas.

ONE CANVAS AND ONE AXES

CODE WALKTHROUGH W05-L07-S03

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig, ax = plt.subplots(figsize=(12, 6.75))
06  ax.bar(["A", "B", "C"], [100, 150, 80])
07  ax.set(title="Branch revenue · January",
08         xlabel="Branch", ylabel="Revenue (EUR)")
09  fig.tight_layout()
10  fig.savefig("figure.png", dpi=320)
11  plt.close(fig)
12  print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

One canvas and one axes

PREDICT THE RESULT

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

ONE CANVAS AND ONE AXES · RESULT

CODE WALKTHROUGH W05-L07-S04

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 plt.rcParams.update({"font.size": 16})
05 fig, ax = plt.subplots(figsize=(12, 6.75))
06 ax.bar(["A", "B", "C"], [100, 150, 80])
07 ax.set(title="Branch revenue · January",
08        xlabel="Branch", ylabel="Revenue (EUR)")
09 fig.tight_layout()
10 fig.savefig("figure.png", dpi=320)
11 plt.close(fig)
12 print("Saved figure.png: 3840 x 2160")
```

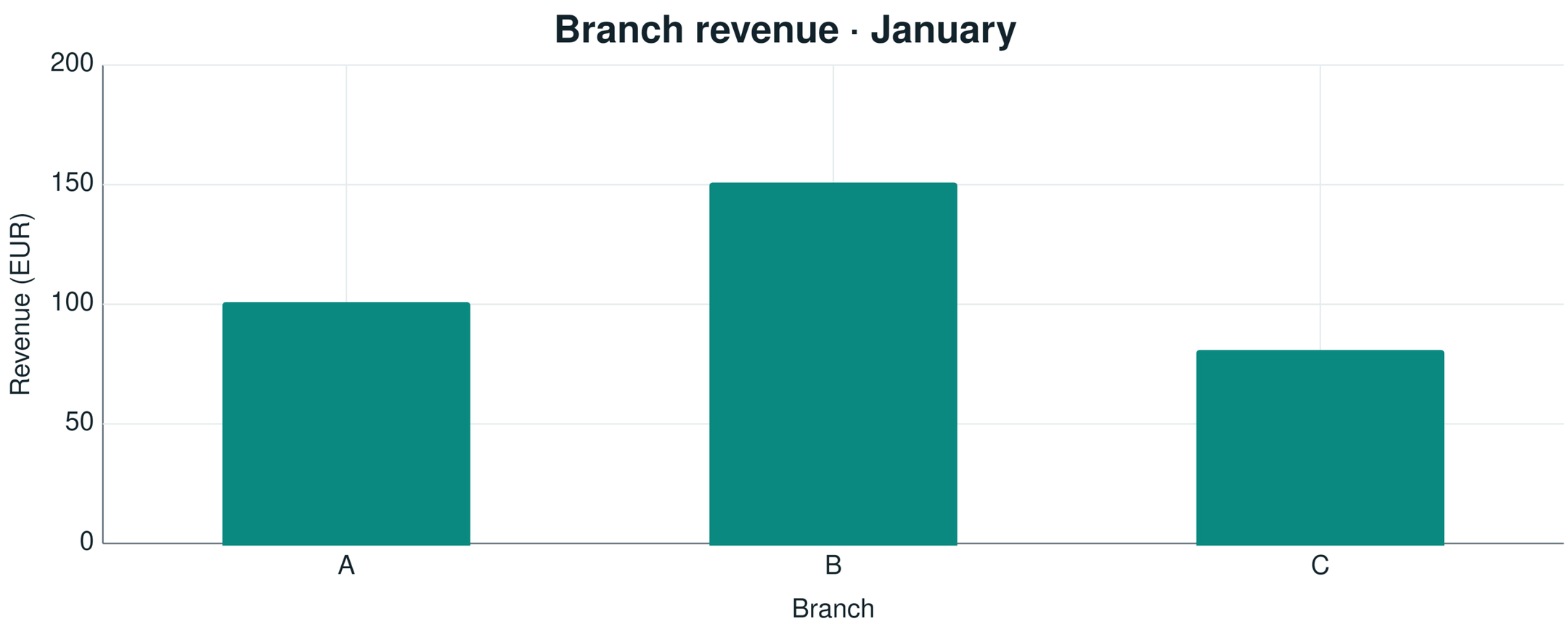
WHAT TO TRACK

One canvas and one axes

VALIDATE THE RESULT

```
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT



THE QUESTION DETERMINES THE MEASURE

CODE WALKTHROUGH W05-L07-S06

```
01  revenue = {"A": 100, "B": 150, "C": 80}
02  winner = max(revenue, key=revenue.get)
03  print("Largest revenue:", winner)
04  print("Coverage:", len(revenue), "branches, January")
```

WHAT TO TRACK

The question determines the measure

PREDICT THE RESULT

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

THE QUESTION DETERMINES THE MEASURE · RESULT

CODE WALKTHROUGH W05-L07-S07

```
01  revenue = {"A": 100, "B": 150, "C": 80}
02  winner = max(revenue, key=revenue.get)
03  print("Largest revenue:", winner)
04  print("Coverage:", len(revenue), "branches, January")
```

WHAT TO TRACK

The question determines the measure

VALIDATE THE RESULT

```
Largest revenue: B
Coverage: 3 branches,
January
```

MISMATCHED PLOT INPUTS

CODE WALKTHROUGH W05-L07-S08

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig, ax = plt.subplots()
06  ax.bar(["A", "B"], [100, 150, 80])
```

WHAT TO TRACK

Mismatched plot inputs

PREDICT THE RESULT

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

MISMATCHED PLOT INPUTS · RESULT

CODE WALKTHROUGH W05-L07-S09

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig, ax = plt.subplots()
06  ax.bar(["A", "B"], [100, 150, 80])
```

WHAT TO TRACK

Mismatched plot inputs

VALIDATE THE RESULT

```
ValueError: shape mismatch:
objects cannot be broadcast
to a single shape.  Mismatch
is between arg 0 with shape
(2,) and arg 1 with shape
(3,).
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Create a bar chart for branches X and Y with revenue 40 and 60 EUR. Label both axes and the period.

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 PRACTICE

LEARNER PRACTICE

Independent practice

State a question about units sold, then draw three products with 2, 5 and 3 units. Keep revenue out of the axis label.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does a larger revenue bar establish higher profit? No. Profit also requires a stated cost measure.

RECAP AND NEXT STEP

Recap and next step

Write the analytical question before plotting. Use fig for the canvas, ax for the plot, and explicit labels for period and units.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 8

Line charts and bar charts

Lines show ordered observations; bars compare separate categories.

THE MENTAL MODEL

MENTAL MODEL

The mental model

Lines show ordered observations; bars compare separate categories.

ORDERED MONTHLY OBSERVATIONS

CODE WALKTHROUGH W05-L08-S03

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig, ax = plt.subplots(figsize=(12,6.75))
06  ax.plot(["Jan","Feb","Mar"],[100,120,110],marker="o")
07  ax.set(title="Monthly revenue · 2025 Q1",
08         xlabel="Month",ylabel="Revenue (EUR)")
09  fig.tight_layout()
10  fig.savefig("figure.png", dpi=320)
11  plt.close(fig)
12  print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

Ordered monthly observations

PREDICT THE RESULT

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

ORDERED MONTHLY OBSERVATIONS · RESULT

CODE WALKTHROUGH W05-L08-S04

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 plt.rcParams.update({"font.size": 16})
05 fig, ax = plt.subplots(figsize=(12,6.75))
06 ax.plot(["Jan","Feb","Mar"],[100,120,110],marker="o")
07 ax.set(title="Monthly revenue · 2025 Q1",
08        xlabel="Month",ylabel="Revenue (EUR)")
09 fig.tight_layout()
10 fig.savefig("figure.png", dpi=320)
11 plt.close(fig)
12 print("Saved figure.png: 3840 x 2160")
```

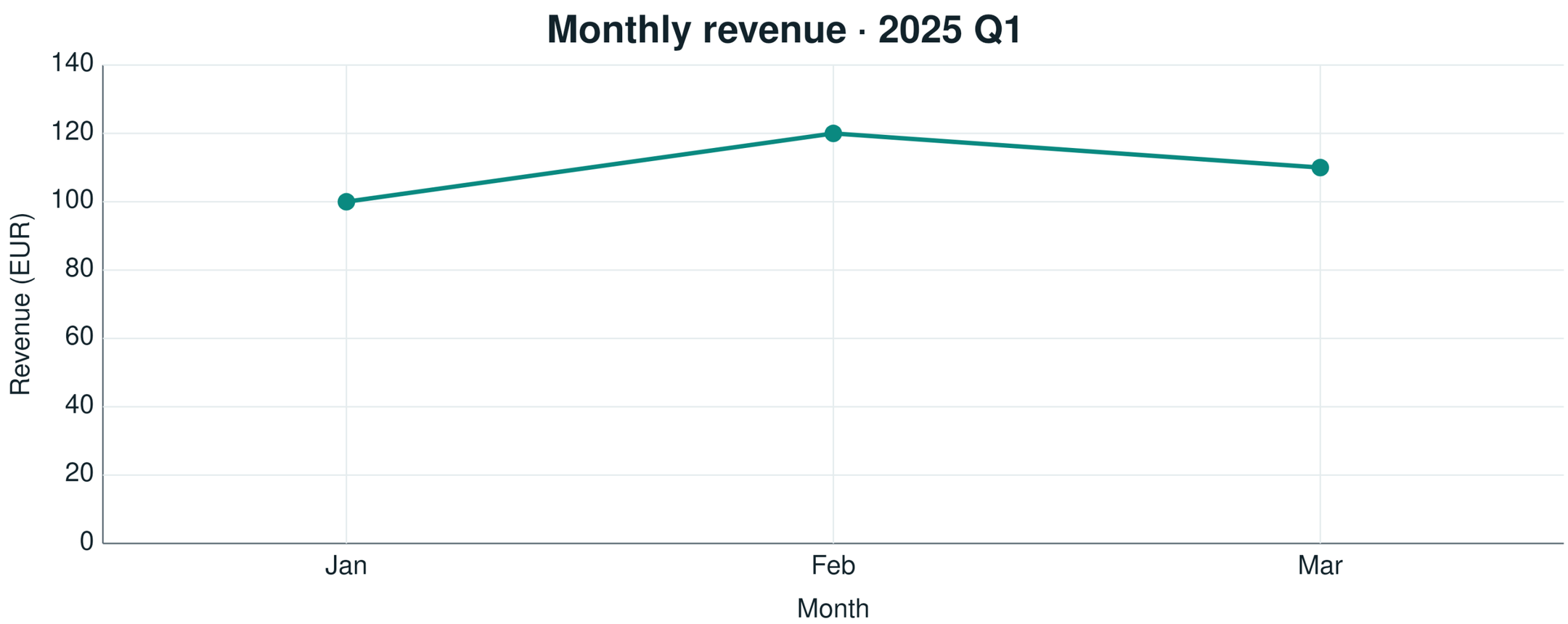
WHAT TO TRACK

Ordered monthly observations

VALIDATE THE RESULT

```
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT



SEPARATE BRANCH CATEGORIES

CODE WALKTHROUGH W05-L08-S06

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 plt.rcParams.update({"font.size": 16})
05 fig, ax = plt.subplots(figsize=(12,6.75))
06 ax.bar(["East","West"],[120,80])
07 ax.set_ylim(bottom=0)
08 ax.set(title="Branch revenue · March",
09        xlabel="Branch",ylabel="Revenue (EUR)")
10 fig.tight_layout()
11 fig.savefig("figure.png", dpi=320)
12 plt.close(fig)
13 print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

Separate branch categories

PREDICT THE RESULT

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

SEPARATE BRANCH CATEGORIES · RESULT

CODE WALKTHROUGH W05-L08-S07

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 plt.rcParams.update({"font.size": 16})
05 fig, ax = plt.subplots(figsize=(12,6.75))
06 ax.bar(["East","West"],[120,80])
07 ax.set_ylim(bottom=0)
08 ax.set(title="Branch revenue · March",
09        xlabel="Branch",ylabel="Revenue (EUR)")
10 fig.tight_layout()
11 fig.savefig("figure.png", dpi=320)
12 plt.close(fig)
13 print("Saved figure.png: 3840 x 2160")
```

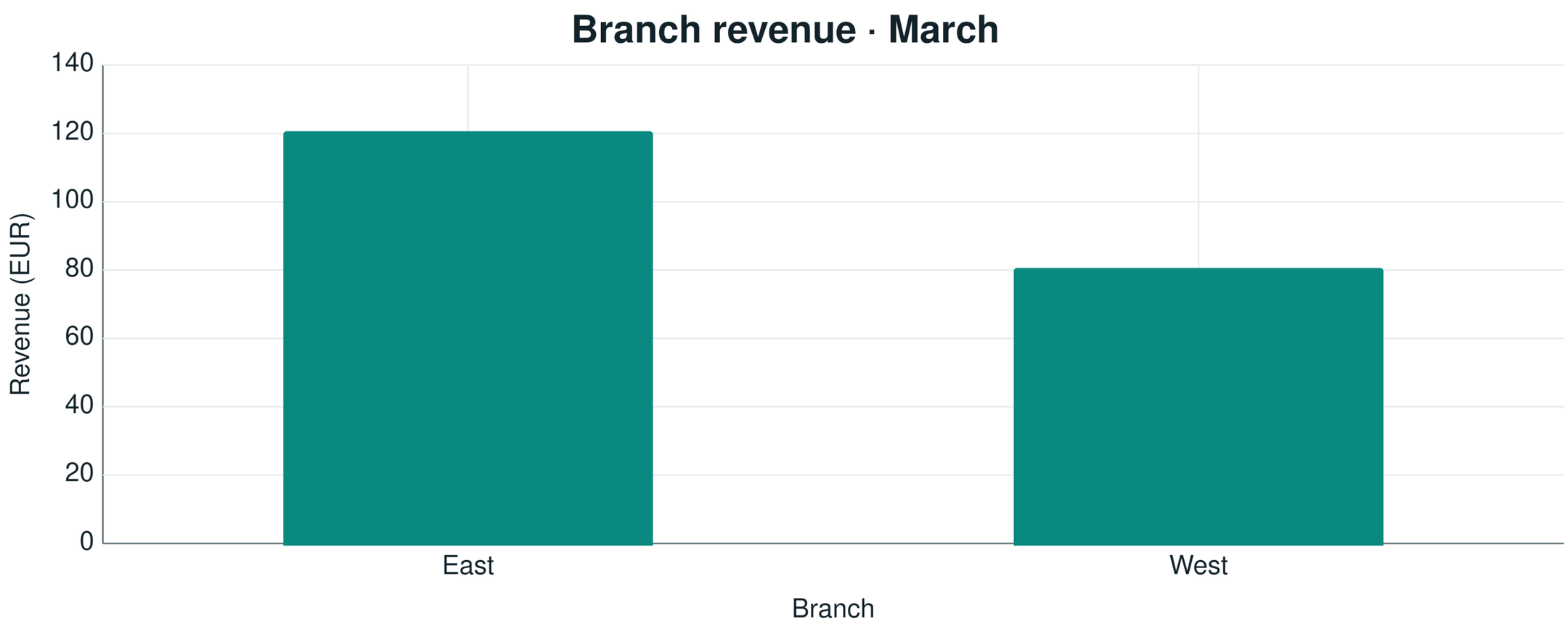
WHAT TO TRACK

Separate branch categories

VALIDATE THE RESULT

```
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT



LEXICAL ORDER IS NOT CALENDAR ORDER

CODE WALKTHROUGH W05-L08-S09

```
01  months = ["Mar", "Jan", "Feb"]
02  order = {"Jan": 1, "Feb": 2, "Mar": 3}
03  print("Alphabetical:", sorted(months))
04  print("Calendar:", sorted(months, key=order.get))
```

WHAT TO TRACK

Lexical order is not calendar order

PREDICT THE RESULT

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

LEXICAL ORDER IS NOT CALENDAR ORDER · RESULT

CODE WALKTHROUGH W05-L08-S10

```
01 months = ["Mar", "Jan", "Feb"]
02 order = {"Jan": 1, "Feb": 2, "Mar": 3}
03 print("Alphabetical:", sorted(months))
04 print("Calendar:", sorted(months, key=order.get))
```

WHAT TO TRACK

Lexical order is not calendar order

VALIDATE THE RESULT

```
Alphabetical: ['Feb', 'Jan',
'Mar']
Calendar: ['Jan', 'Feb',
'Mar']
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Plot chronological months Jan, Feb, Mar with units 3, 7, 5. Use markers and a units label.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Draw a bar chart for products A and B with counts 10 and 20. Preserve a zero baseline.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Should East and West be connected to suggest a monthly trend? No. They are distinct categories in this question.

RECAP AND NEXT STEP

Recap and next step

Sort time before connecting observations. Use bars for categories, markers for observations, and clearly labelled scales.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 9

Distributions, histograms and box plots

A distribution describes how values are spread, including unusual observations.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A distribution describes how values are spread, including unusual observations.

HISTOGRAM WITH STATED BINS

CODE WALKTHROUGH W05-L09-S03

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  values=[10,11,12,13,100]
06  fig,ax=plt.subplots(figsize=(12,6.75))
07  ax.hist(values,bins=[0,20,40,60,80,120],edgecolor="black")
08  ax.set(title="Invoice distribution · 5 observations",
09         xlabel="Invoice amount (EUR)",ylabel="Invoice count")
10  fig.tight_layout()
11  fig.savefig("figure.png", dpi=320)
12  plt.close(fig)
13  print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

Histogram with stated bins

PREDICT THE RESULT

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

HISTOGRAM WITH STATED BINS · RESULT

CODE WALKTHROUGH W05-L09-S04

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  values=[10,11,12,13,100]
06  fig,ax=plt.subplots(figsize=(12,6.75))
07  ax.hist(values,bins=[0,20,40,60,80,120],edgecolor="black")
08  ax.set(title="Invoice distribution · 5 observations",
09         xlabel="Invoice amount (EUR)",ylabel="Invoice count")
10  fig.tight_layout()
11  fig.savefig("figure.png", dpi=320)
12  plt.close(fig)
13  print("Saved figure.png: 3840 x 2160")
```

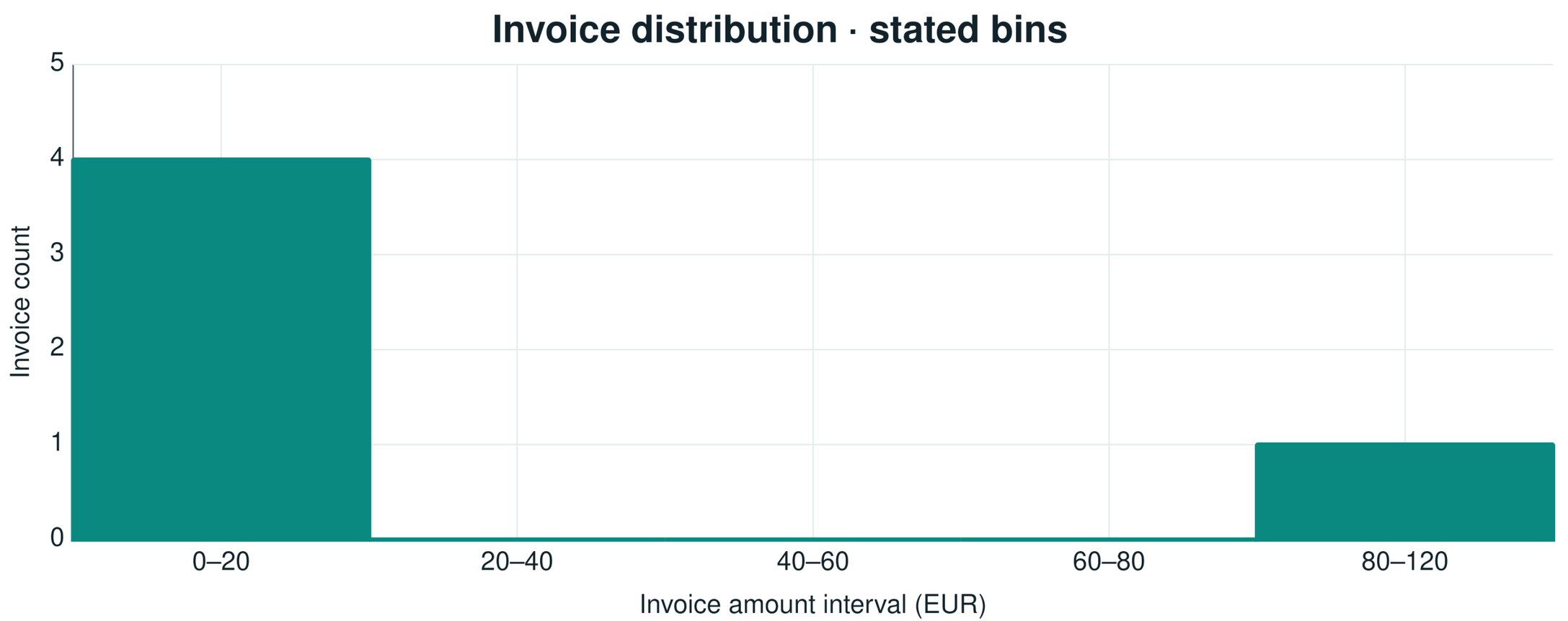
WHAT TO TRACK

Histogram with stated bins

VALIDATE THE RESULT

```
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT



BOX PLOT AND ROBUST SUMMARY

CODE WALKTHROUGH W05-L09-S06

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  import pandas as pd
06  values=[10,11,12,13,100]
07  fig,ax=plt.subplots(figsize=(12,6.75))
08  ax.boxplot(values,orientation="horizontal")
09  ax.set(title="Invoice amounts · extreme value retained",
10         xlabel="Invoice amount (EUR)")
11  print("Median:",pd.Series(values).median())
12  print("Mean:",pd.Series(values).mean())
13  fig.tight_layout()
14  fig.savefig("figure.png", dpi=320)
15  plt.close(fig)
16  print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

Box plot and robust summary

PREDICT THE RESULT

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

BOX PLOT AND ROBUST SUMMARY · RESULT

CODE WALKTHROUGH W05-L09-S07

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 plt.rcParams.update({"font.size": 16})
05 import pandas as pd
06 values=[10,11,12,13,100]
07 fig,ax=plt.subplots(figsize=(12,6.75))
08 ax.boxplot(values,orientation="horizontal")
09 ax.set(title="Invoice amounts · extreme value retained",
10        xlabel="Invoice amount (EUR)")
11 print("Median:",pd.Series(values).median())
12 print("Mean:",pd.Series(values).mean())
13 fig.tight_layout()
14 fig.savefig("figure.png", dpi=320)
15 plt.close(fig)
16 print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

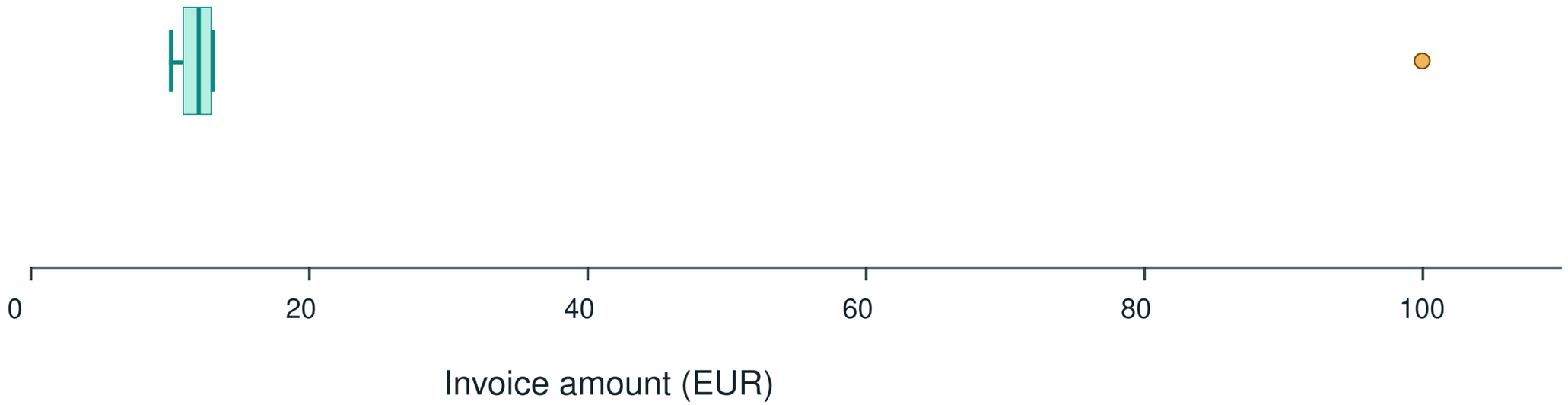
Box plot and robust summary

VALIDATE THE RESULT

```
Median: 12.0
Mean: 29.2
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT

Invoice amounts · extreme value retained



Q1=11 · Median=12 · Q3=13 · Whiskers=10,13 · Outlier=100 retained

ONE OBSERVATION IS INSUFFICIENT FOR SPREAD

CODE WALKTHROUGH W05-L09-S09

```
01  import pandas as pd
02  s=pd.Series([12.0])
03  print("Count:",s.count())
04  print("Median:",s.median())
05  print("Sample SD:",s.std(ddof=1))
```

WHAT TO TRACK

One observation is insufficient for spread

PREDICT THE RESULT

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

ONE OBSERVATION IS INSUFFICIENT FOR SPREAD · RESULT

CODE WALKTHROUGH W05-L09-S10

```
01  import pandas as pd
02  s=pd.Series([12.0])
03  print("Count:",s.count())
04  print("Median:",s.median())
05  print("Sample SD:",s.std(ddof=1))
```

WHAT TO TRACK

One observation is insufficient for spread

VALIDATE THE RESULT

```
Count: 1
Median: 12.0
Sample SD: nan
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

For [2,3,4,5,50], compute mean, median, first and third quartiles. Keep every observation.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Create a histogram for [1,1,2,3,8] with bin boundaries [0,2,4,6,10]. Label counts and measurement units.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does a box-plot outlier prove a data error? No. It is a rule-based flag that needs contextual investigation.

RECAP AND NEXT STEP

Recap and next step

Inspect distributions, state histogram bins, compare mean with median, and investigate extreme observations before changing them.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 10

Scatter plots, labels and annotations

A scatter plot preserves paired observations; association does not establish a cause.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A scatter plot preserves paired observations; association does not establish a cause.

PAIRED OBSERVATIONS

CODE WALKTHROUGH W05-L10-S03

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig,ax=plt.subplots(figsize=(12,6.75))
06  ax.scatter([10,20,30,40],[50,65,62,90])
07  ax.set(title="Four fictional branches · March",
08         xlabel="Advertising spend (EUR)",ylabel="Sales (units)")
09  fig.tight_layout()
10  fig.savefig("figure.png", dpi=320)
11  plt.close(fig)
12  print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

Paired observations

PREDICT THE RESULT

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

PAIRED OBSERVATIONS · RESULT

CODE WALKTHROUGH W05-L10-S04

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig,ax=plt.subplots(figsize=(12,6.75))
06  ax.scatter([10,20,30,40],[50,65,62,90])
07  ax.set(title="Four fictional branches · March",
08         xlabel="Advertising spend (EUR)",ylabel="Sales (units)")
09  fig.tight_layout()
10  fig.savefig("figure.png", dpi=320)
11  plt.close(fig)
12  print("Saved figure.png: 3840 x 2160")
```

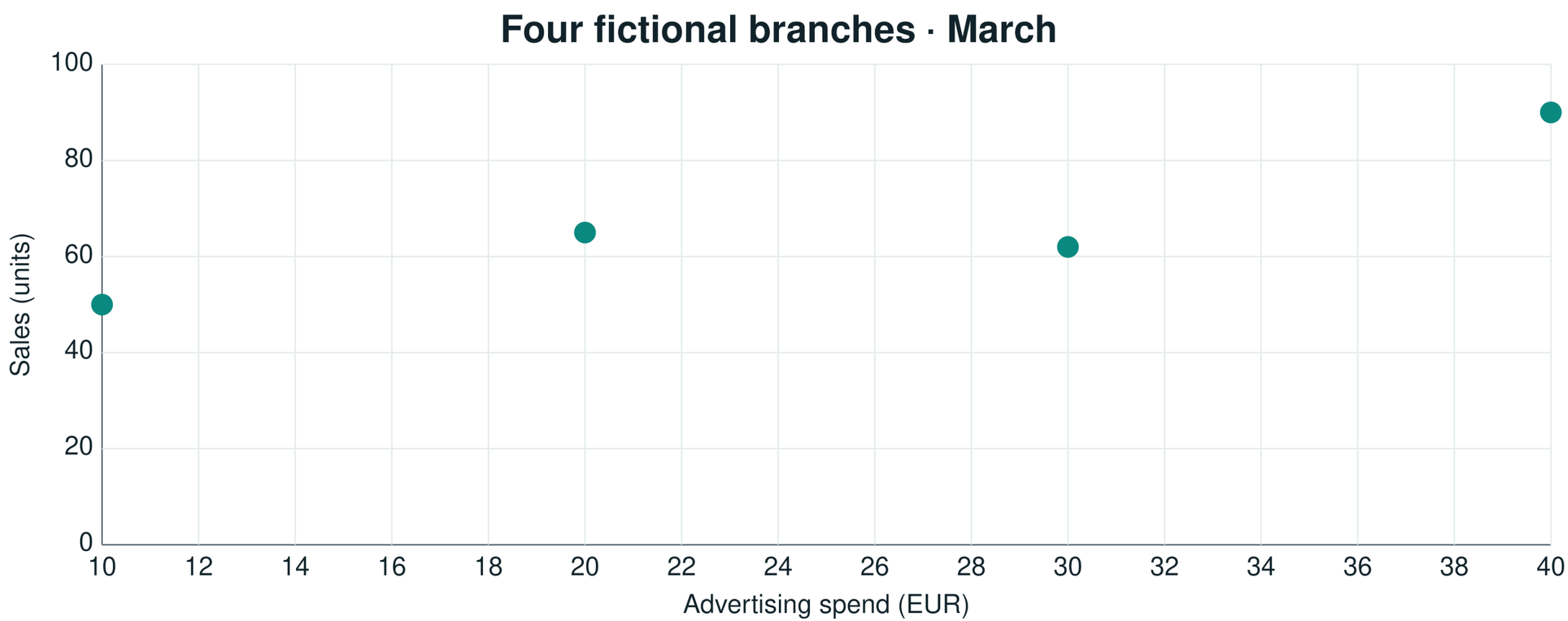
WHAT TO TRACK

Paired observations

VALIDATE THE RESULT

```
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT



ANNOTATE A KNOWN OBSERVATION

CODE WALKTHROUGH W05-L10-S06

```
01  import matplotlib
02  matplotlib.use("Agg")
03  import matplotlib.pyplot as plt
04  plt.rcParams.update({"font.size": 16})
05  fig,ax=plt.subplots(figsize=(12,6.75))
06  ax.scatter([10,20,30],[50,65,62])
07  ax.annotate("Branch C",xy=(30,62),xytext=(24,52),
08              arrowprops={"arrowstyle":"->"})
09  ax.set(title="Observed branch comparison · March",
10         xlabel="Spend (EUR)",ylabel="Sales (units)")
11  fig.tight_layout()
12  fig.savefig("figure.png", dpi=320)
13  plt.close(fig)
14  print("Saved figure.png: 3840 x 2160")
```

WHAT TO TRACK

Annotate a known observation

PREDICT THE RESULT

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

ANNOTATE A KNOWN OBSERVATION · RESULT

CODE WALKTHROUGH W05-L10-S07

```
01 import matplotlib
02 matplotlib.use("Agg")
03 import matplotlib.pyplot as plt
04 plt.rcParams.update({"font.size": 16})
05 fig,ax=plt.subplots(figsize=(12,6.75))
06 ax.scatter([10,20,30],[50,65,62])
07 ax.annotate("Branch C",xy=(30,62),xytext=(24,52),
08             arrowprops={"arrowstyle":"->"})
09 ax.set(title="Observed branch comparison · March",
10        xlabel="Spend (EUR)",ylabel="Sales (units)")
11 fig.tight_layout()
12 fig.savefig("figure.png", dpi=320)
13 plt.close(fig)
14 print("Saved figure.png: 3840 x 2160")
```

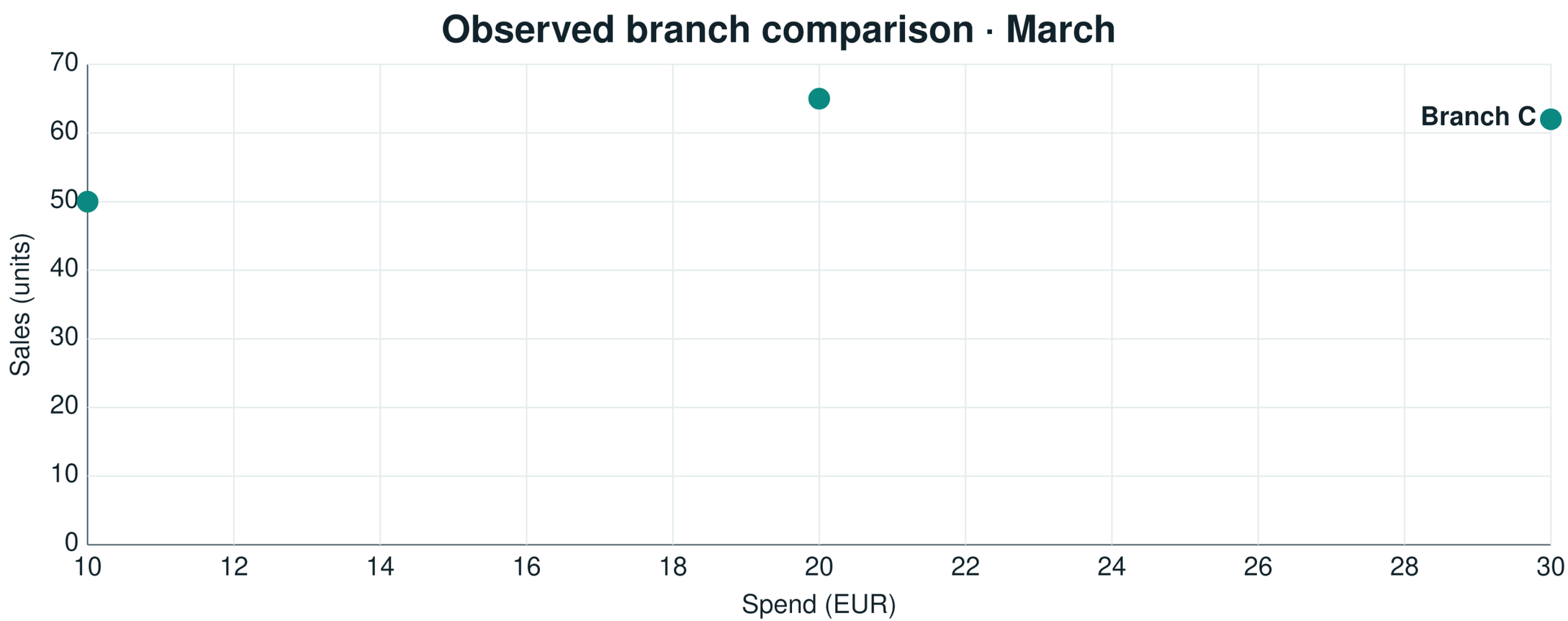
WHAT TO TRACK

Annotate a known observation

VALIDATE THE RESULT

```
Saved figure.png: 3840 x
2160
```

VISIBLE CHART RESULT



INDEPENDENTLY SORTED PAIRS ARE WRONG

CODE WALKTHROUGH W05-L10-S09

```
01  pairs=[(20,65),(10,50),(30,62)]
02  print("Correct paired sort:",sorted(pairs))
03  print("Incorrect recombination:",list(zip(
04      sorted(x for x,y in pairs),
05      sorted(y for x,y in pairs))))
```

WHAT TO TRACK

Independently sorted pairs are wrong

PREDICT THE RESULT

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

INDEPENDENTLY SORTED PAIRS ARE WRONG · RESULT

CODE WALKTHROUGH W05-L10-S10

```
01  pairs=[(20,65),(10,50),(30,62)]
02  print("Correct paired sort:",sorted(pairs))
03  print("Incorrect recombination:",list(zip(
04      sorted(x for x,y in pairs),
05      sorted(y for x,y in pairs))))
```

WHAT TO TRACK

Independently sorted pairs are wrong

VALIDATE THE RESULT

```
Correct paired sort: [(10,
50), (20, 65), (30, 62)]
Incorrect recombination:
[(10, 50), (20, 62), (30,
65)]
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Plot paired quantities [(1,4),(2,3),(3,8)] with horizontal hours and vertical units. Preserve each pair.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Annotate the point (3,8) as Trial C. State that these three observations do not establish causation.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does independently sorting both columns preserve their relationship? No.
It can attach one entity’s value to another entity’s value.

RECAP AND NEXT STEP

Recap and next step

Preserve pairs, label units and coverage, and use annotations to identify observations. Describe association cautiously.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 11

Descriptive statistics and explicit conventions

A statistic summarises a dataset under an explicit counting and interpolation convention.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A statistic summarises a dataset under an explicit counting and interpolation convention.

MEAN, MEDIAN AND TWO SD CONVENTIONS

CODE WALKTHROUGH W05-L11-S03

```
01  import pandas as pd
02  s=pd.Series([10.0,20.0,30.0])
03  print("Mean:",s.mean(),"Median:",s.median())
04  print("Population SD:",round(s.std(ddof=0),3))
05  print("Sample SD:",s.std(ddof=1))
```

WHAT TO TRACK

Mean, median and two SD conventions

PREDICT THE RESULT

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

MEAN, MEDIAN AND TWO SD CONVENTIONS · RESULT

CODE WALKTHROUGH W05-L11-S04

```
01  import pandas as pd
02  s=pd.Series([10.0,20.0,30.0])
03  print("Mean:",s.mean(),"Median:",s.median())
04  print("Population SD:",round(s.std(ddof=0),3))
05  print("Sample SD:",s.std(ddof=1))
```

WHAT TO TRACK

Mean, median and two SD conventions

VALIDATE THE RESULT

```
Mean: 20.0 Median: 20.0
Population SD: 8.165
Sample SD: 10.0
```

QUANTILES WITH LINEAR INTERPOLATION

CODE WALKTHROUGH W05-L11-S05

```
01  import pandas as pd
02  s=pd.Series([0,10,20,30])
03  print(s.quantile([.25,.5,.75],interpolation="linear"))
04  print("Observed:",s.count())
```

WHAT TO TRACK

Quantiles with linear interpolation

PREDICT THE RESULT

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

QUANTILES WITH LINEAR INTERPOLATION · RESULT

CODE WALKTHROUGH W05-L11-S06

```
01  import pandas as pd
02  s=pd.Series([0,10,20,30])
03  print(s.quantile([.25,.5,.75],interpolation="linear"))
04  print("Observed:",s.count())
```

WHAT TO TRACK

Quantiles with linear interpolation

VALIDATE THE RESULT

```
0.25     7.5
0.50    15.0
0.75    22.5
dtype: float64
Observed: 4
```

MISSING OBSERVATIONS CHANGE THE DENOMINATOR

CODE WALKTHROUGH W05-L11-S07

```
01  import pandas as pd
02  s=pd.Series([10.0,None,30.0])
03  print("Rows:",len(s),"Observed:",s.count())
04  print("Observed mean:",s.mean())
05  print("Sample variance:",s.var(ddof=1))
```

WHAT TO TRACK

Missing observations change the denominator

PREDICT THE RESULT

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

MISSING OBSERVATIONS CHANGE THE DENOMINATOR · RESULT

CODE WALKTHROUGH W05-L11-S08

```
01  import pandas as pd
02  s=pd.Series([10.0,None,30.0])
03  print("Rows:",len(s),"Observed:",s.count())
04  print("Observed mean:",s.mean())
05  print("Sample variance:",s.var(ddof=1))
```

WHAT TO TRACK

Missing observations change the denominator

VALIDATE THE RESULT

```
Rows: 3 Observed: 2
Observed mean: 20.0
Sample variance: 200.0
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

For [2,4,6], calculate mean, sample variance and sample SD with ddof=1.

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 PRACTICE

LEARNER PRACTICE

Independent practice

For [0,20], compute the 25th percentile using linear and nearest interpolation and explain their difference.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Must sample SD equal population SD on the same values? No. Their denominators differ when ddof differs.

RECAP AND NEXT STEP

Recap and next step

Report counts, define the statistic, state ddof and quantile interpolation, and retain measurement units.

UNDERSTAND APPLY VALIDATE EXPLAIN

FOUNDATION LESSON

WEEK 5 · LESSON 12

Correlation and a decision-ready evidence pack

A correlation summarises paired linear association; an evidence pack preserves how the result was produced.

THE MENTAL MODEL

MENTAL MODEL

The mental model

A correlation summarises paired linear association; an evidence pack preserves how the result was produced.

A SIMPLE PAIRED CORRELATION

CODE WALKTHROUGH W05-L12-S03

```
01  import pandas as pd
02  d=pd.DataFrame({"hours":[1,2,3,4],"units":[2,4,6,8]})
03  print("Pairs:",len(d.dropna()))
04  print("Pearson r:",d["hours"].corr(d["units"]))
```

WHAT TO TRACK

A simple paired correlation

PREDICT THE RESULT

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

A SIMPLE PAIRED CORRELATION · RESULT

CODE WALKTHROUGH W05-L12-S04

```
01  import pandas as pd
02  d=pd.DataFrame({"hours":[1,2,3,4],"units":[2,4,6,8]})
03  print("Pairs:",len(d.dropna()))
04  print("Pearson r:",d["hours"].corr(d["units"]))
```

WHAT TO TRACK

A simple paired correlation

VALIDATE THE RESULT

```
Pairs: 4
Pearson r: 1.0
```

A CONCISE EVIDENCE PACK

CODE WALKTHROUGH W05-L12-S05

```
01  import pandas as pd
02  from pathlib import Path
03  d=pd.DataFrame({"branch":["A","B"],"revenue_eur":[120,80]})
04  out=Path("outputs");out.mkdir(exist_ok=True)
05  d.to_csv(out/"summary.csv",index=False,encoding="utf-8")
06  finding="March: A recorded EUR 120 and B EUR 80. Costs unavaila
↪   ble."
07  (out/"findings.txt").write_text(finding,encoding="utf-8")
08  print(finding)
09  print("Rows:",len(d),"Total EUR:",d["revenue_eur"].sum())
```

WHAT TO TRACK

A concise evidence pack

PREDICT THE RESULT

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

A CONCISE EVIDENCE PACK · RESULT

CODE WALKTHROUGH W05-L12-S06

```
01  import pandas as pd
02  from pathlib import Path
03  d=pd.DataFrame({"branch":["A","B"],"revenue_eur":[120,80]})
04  out=Path("outputs");out.mkdir(exist_ok=True)
05  d.to_csv(out/"summary.csv",index=False,encoding="utf-8")
06  finding="March: A recorded EUR 120 and B EUR 80. Costs unavaila
↪   ble."
07  (out/"findings.txt").write_text(finding,encoding="utf-8")
08  print(finding)
09  print("Rows:",len(d),"Total EUR:",d["revenue_eur"].sum())
```

WHAT TO TRACK

A concise evidence pack

VALIDATE THE RESULT

```
March: A recorded EUR 120
and B EUR 80. Costs
unavailable.
Rows: 2 Total EUR: 200
```

A CURVED RELATIONSHIP WITH ZERO CORRELATION

CODE WALKTHROUGH W05-L12-S07

```
01  import pandas as pd
02  x=pd.Series([-2,-1,0,1,2],dtype="float64")
03  y=x*x
04  print("Pearson r:",round(x.corr(y),6))
05  print("Pairs:",list(zip(x.tolist(),y.tolist())))
```

WHAT TO TRACK

A curved relationship with zero correlation

PREDICT THE RESULT

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

A CURVED RELATIONSHIP WITH ZERO CORRELATION · RESULT

CODE WALKTHROUGH W05-L12-S08

```
01  import pandas as pd
02  x=pd.Series([-2,-1,0,1,2],dtype="float64")
03  y=x*x
04  print("Pearson r:",round(x.corr(y),6))
05  print("Pairs:",list(zip(x.tolist(),y.tolist())))
```

WHAT TO TRACK

A curved relationship with zero correlation

VALIDATE THE RESULT

```
Pearson r: 0.0
Pairs: [(-2.0, 4.0), (-1.0,
1.0), (0.0, 0.0), (1.0,
1.0), (2.0, 4.0)]
```

GUIDED PRACTICE

LEARNER PRACTICE

Guided practice

Calculate correlation for pairs (1,3), (2,6), (3,9). State the paired count and one causal limitation.

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 PRACTICE

LEARNER PRACTICE

Independent practice

Create a summary for branches X/Y with revenue 90/60 EUR, save it as UTF-8 CSV, and write a finding limited to revenue.

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.

CHECK YOUR UNDERSTANDING

MENTAL MODEL

Check your understanding

Diagnostic: Does zero Pearson correlation prove no relationship? No. A curved relationship can have zero linear correlation.

RECAP AND NEXT STEP

Recap and next step

Preserve paired observations and inspect their shape. Deliver rules, counts, tables, charts and evidence-backed findings with limitations.

UNDERSTAND APPLY VALIDATE EXPLAIN