

AURXON ENGINEERING NOTES

B.Tech AI/ML • Semester I • Foundations → Practice → Confidence

EXPLORATORY DATA ANALYSIS

A first-semester reasoning-first textbook + practical workbook

Designed for different starting points.

Start from zero, build the Python and statistics intuition you need, practise every idea on tiny data, then move toward real datasets, exams, internships and entry-level data/ML work.

The learning loop: Question → Observe → Measure → Visualise → Explain → Decide → Re-check.

Karann • AURXON

Theory • Code • Labs • Exams • Interview
Thinking

Where Intelligence Meets Execution

How to Use This Book

EDA is not a collection of plots. It is a disciplined way of asking: *What does this dataset actually say, what might be misleading us, and what should we do next?*

Every major topic in this book follows the same learning loop:

1. **Understand the object.** What does a row represent? What does a column measure? What is the unit and time window?
2. **Observe before changing.** Profile types, missingness, ranges, duplicates, distributions and relationships.
3. **Form a hypothesis.** State what you expect and why.
4. **Test with a small, interpretable calculation.** Prefer a simple statistic or plot before a complicated model.
5. **Decide and document.** Keep, transform, impute, remove, encode, or investigate—with a reason.
6. **Re-check after every important transformation.** A clean-looking table can still be wrong.

Think Like an Engineer

The central skill is not memorising pandas commands. The goal is to become the person who can receive an unfamiliar CSV at 10:00 AM, ask the right questions by 10:10, find a suspicious pattern by 10:30, explain it without overclaiming by 11:00, and leave behind reproducible code that another engineer can audit.

How to study each chapter

For every concept, do three passes: (1) explain it aloud without code, (2) reproduce the code from memory on a tiny dataset, and (3) change the dataset and predict what should happen before running the cell. The third pass is where real reasoning develops.

Syllabus Map — What You Must Actually Learn

The university syllabus is the spine of this book. Advanced material is included only when it helps a student understand, practise, answer an exam question, or become more capable in a real project.

Unit	Core syllabus	What you will be able to do
1	Introduction to EDA; steps; importance; data types, structure and distributions; Jupyter/Colab; Pandas basics	Open an unfamiliar dataset, describe its structure, classify variables, inspect distributions and write safe first-pass Pandas code.
2	Central tendency and dispersion; histograms, box plots, scatter plots, pair plots; outliers and missing values	Choose appropriate summaries/plots, calculate and interpret statistics, identify suspicious observations and explain what a visual actually shows.
3	Missing, duplicate and inconsistent data; normalization, binning, encoding; feature engineering and selection	Build a defensible cleaning workflow, transform data without accidental leakage, create useful features and justify selection decisions.
4	Correlation and multivariate relationships; PCA; EDA reports and dashboards	Investigate relationships, understand correlation limits, explain PCA conceptually, and communicate findings in a report/dashboard.
5	Healthcare, retail, finance; complete EDA pipeline on real-world data	Adapt EDA to domain context and deliver an end-to-end, reproducible project with evidence, limitations and recommendations.

Intuition & Real-World Analogy

Do not compare yourself with another student's starting point. Some students will already know Python; others will be seeing a DataFrame for the first time. The book therefore uses three levels repeatedly: **Start Here** (minimum understanding), **Understand the Logic** (why it works), and **Apply It** (independent problem solving). You should be able to move upward at your own speed without skipping the foundation.

The three-level learning system

START HERE

Tiny examples, plain language, one concept at a time. If you are weak in coding, do not rush this level.

UNDERSTAND

Derivations, comparisons, edge cases and interpretation. Explain the idea without looking at the code.

APPLY

Unseen datasets, debugging, design questions, practical labs and interview-style tasks. This is where memorisation stops helping.

Think Like an Engineer

A rule for this book: predict before you execute. Before running a code cell, write one sentence about what you expect. After execution, compare the output with your prediction. If they differ, investigate. That small habit builds programming and analytical reasoning much faster than repeatedly copying cells.

What counts as mastery?

A student has not mastered a topic merely because they can reproduce a function call. Mastery means they can: (1) identify the question, (2) choose an appropriate method, (3) implement it, (4) interpret the result, (5) identify a limitation, and (6) explain what they would do next.

Contents

How to Use This Book	1
Syllabus Map — What You Must Actually Learn	2
1 Unit 1 — Foundations of EDA & Environment Setup	11
1.1 The question behind EDA	11
1.2 EDA versus data cleaning versus modelling	11
1.3 The complete EDA lifecycle	12
1.3.1 Step 0 — Define the unit of observation	12
1.3.2 Step 1 — Establish provenance	12
1.3.3 Step 2 — Profile structure before statistics	12
1.3.4 Step 3 — Form hypotheses before hunting patterns	13
1.4 Data types: measurement scale, not just Python dtype	13
1.4.1 Nominal	13
1.4.2 Ordinal	13
1.4.3 Interval	13
1.4.4 Ratio	13
1.4.5 Discrete versus continuous	13
1.5 Structural layouts	14
1.5.1 Wide versus long	14
1.5.2 Tidy-data principles	14
1.6 Statistical distributions: learn the shape before the formula	14
1.6.1 Common distribution shapes	14
1.7 Python environment: reproducibility before exploration	14
1.7.1 Jupyter and Google Colab workflow	14
1.7.2 Dependency management	15
1.8 Pandas fundamentals with reasoning	15
1.8.1 Series	15
1.8.2 DataFrame	15
1.8.3 Indexing: label versus position	15
1.8.4 Vectorisation	16
1.9 Unit 1 practical build: an EDA starter function	16
1.9.1 Logic-building lab	17
1.10 Probability and distribution toolkit	17
1.10.1 Population versus sample	17
1.10.2 Sampling variability	17
1.10.3 Sampling bias	17
1.10.4 Bernoulli distribution	17
1.10.5 Binomial distribution	17
1.10.6 Poisson distribution	18
1.10.7 Uniform distribution	18
1.10.8 Normal distribution	18
1.10.9 Exponential distribution	19
1.10.10 Student's t and chi-square distributions	19

1.10.11	Why distribution knowledge helps EDA	19
1.11	Data-generating process thinking	19
1.12	Measurement error and units	19
2	Unit 2 — Descriptive Statistics & Visual Discovery	21
2.1	The discipline of summarisation	21
2.2	Central tendency	21
2.2.1	Arithmetic mean	21
2.2.2	Median	21
2.2.3	Mode	22
2.2.4	Trimmed mean	22
2.3	Dispersion	22
2.3.1	Sample variance and standard deviation	22
2.3.2	IQR	22
2.3.3	MAD	22
2.3.4	Coefficient of variation	22
2.4	Skewness and kurtosis	23
2.5	Visual discovery	23
2.5.1	Histogram	23
2.5.2	KDE	23
2.5.3	Box plot	23
2.5.4	Violin plot	23
2.5.5	Scatter plot	23
2.5.6	Pair plot	24
2.6	Univariate, bivariate and multivariate reasoning	24
2.7	Outlier detection	24
2.7.1	Z-score	24
2.7.2	IQR rule	24
2.8	Missingness profiling	24
2.9	Worked reasoning example: the mean lies	25
2.10	Unit 2 practical lab	25
2.11	Unit 2 exercises	25
2.11.1	MCQs and short answers	25
2.11.2	Find the bug	25
3	Unit 3 — Data Cleaning & Feature Engineering	26
3.1	Cleaning is a decision system	26
3.2	Missing data: MCAR, MAR, MNAR	26
3.2.1	MCAR — Missing Completely At Random	26
3.2.2	MAR — Missing At Random	26
3.2.3	MNAR — Missing Not At Random	26
3.2.4	A practical missingness investigation	26
3.2.5	Imputation strategies	27
3.3	Deduplication	27
3.4	Structural inconsistencies	27
3.4.1	Datetime conversion	28
3.4.2	Categoricals	28
3.5	Scaling and transformation	28
3.5.1	Min-Max scaling	28
3.5.2	L2 normalization	28
3.5.3	Z-score standardisation	28
3.5.4	Robust scaling	28
3.5.5	Log and power transforms	28
3.5.6	Discretisation / binning	29
3.6	Categorical encoding	29
3.6.1	One-hot encoding	29

3.6.2	Ordinal encoding	29
3.6.3	Frequency encoding	29
3.6.4	Target encoding	29
3.7	Feature engineering	29
3.7.1	Interaction features	29
3.7.2	Polynomial features	30
3.7.3	Datetime parsing	30
3.7.4	Filter-based metrics	30
3.8	Leakage: the hidden failure mode	30
3.9	Unit 3 integrated cleaning workflow	31
3.10	Unit 3 lab: build a cleaning contract	31
3.11	Unit 3 exercises	31
4	Unit 4 — Multivariate Analysis, PCA & Reporting	32
4.1	Correlation is a lens, not a verdict	32
4.2	Pearson correlation	32
4.3	Spearman correlation	32
4.4	Kendall Tau	32
4.5	Covariance matrix	32
4.6	Multicollinearity and VIF	33
4.7	PCA: principal component analysis	33
4.7.1	Geometric intuition	33
4.7.2	Step 1 — Center the features	34
4.7.3	Step 2 — Form covariance	34
4.7.4	Step 3 — Solve the eigenproblem	34
4.7.5	Step 4 — Rank components	34
4.8	Scree plots and component selection	35
4.9	Automated EDA reports	35
4.10	Streamlit dashboard design	35
4.11	Unit 4 exercises	35
5	Unit 5 — Domain Case Studies & Production Capstone	37
5.1	Why domain context changes EDA	37
5.2	Healthcare EDA	37
5.2.1	Class imbalance	37
5.2.2	Privacy and HIPAA-aware handling	37
5.3	Retail EDA	37
5.3.1	Time seasonality	38
5.3.2	Customer cohorts	38
5.4	Finance EDA	38
5.4.1	Returns	38
5.4.2	Volatility	38
5.4.3	Phase 3 — Reproducible profiling	39
5.4.4	Phase 4 — Distribution and relationship investigation	39
5.4.5	Phase 5 — Cleaning contract	39
5.4.6	Phase 6 — Feature engineering	40
5.4.7	Phase 7 — Validation	40
5.4.8	Phase 8 — Final report	40
5.5	Capstone dataset ladder	40
5.6	Capstone grading rubric	40
5.7	Independent thinking drills	40
6	Cross-Cutting Engineering Skills	42
6.1	How to read a dataset like an engineer	42
6.2	The evidence ladder	42
6.3	Reproducibility checklist	42

6.4	Debugging methodology for EDA code	43
6.5	Exam strategy: answer the reasoning, not just the definition	43
6.6	Master formula sheet	43
6.7	Final challenge: the 90-minute EDA interview simulation	43
7	Deep Practice Workbook — Unit 1	45
7.1	Worked example: from an unknown CSV to a data map	45
7.2	Schema notebook pattern	45
7.3	Row-grain drill	46
7.4	Identifiers versus features	46
7.5	Distribution drill: same mean, different reality	46
7.6	Distribution drill: ECDF	46
7.7	Distribution drill: Q-Q thinking	47
7.8	Unit 1 mini-project: dataset autopsy	47
8	Deep Practice Workbook — Unit 2	48
8.1	Central tendency worked calculation	48
8.2	Variance worked calculation	48
8.3	Why <i>ddof</i> matters	48
8.4	Skewness formula and interpretation	48
8.5	Kurtosis and tails	49
8.6	Plot choice decision tree	49
8.7	Histogram bin sensitivity lab	49
8.8	Box plot versus violin plot	49
8.9	Outlier lab: four cases	49
8.10	Missingness matrix interpretation	50
9	Deep Practice Workbook — Unit 3	51
9.1	Cleaning decision table	51
9.2	Imputation distortion demonstration	51
9.3	Group-wise imputation reasoning	51
9.4	Target encoding: leakage-safe mental model	52
9.5	Encoding selection table	52
9.6	Feature engineering: source-to-feature ledger	52
9.7	Datetime engineering lab	52
10	Deep Practice Workbook — Unit 4	54
10.1	Three correlation worlds	54
10.2	Correlation is not causation lab	54
10.3	Simpson's paradox exercise	54
10.4	VIF worked example	54
10.5	PCA from scratch on a two-dimensional dataset	54
10.6	PCA reconstruction thought experiment	55
10.7	Automated EDA: what the machine can and cannot decide	55
10.8	Dashboard design lab	55
11	Deep Practice Workbook — Unit 5 and Capstone	56
11.1	Healthcare scenario	56
11.2	Retail scenario	56
11.3	Finance scenario	56
11.4	Capstone sprint plan	57
11.5	Capstone failure injection	57
11.6	Final practical checklist	57
11.7	Answer-writing practice	57
11.8	Student self-improvement loop	58

12 Master Question Bank	59
12.1 Conceptual questions	59
12.2 Coding questions	59
12.3 Debugging questions	60
12.4 Design questions	60
13 First-Semester Python Bridge for EDA	61
13.1 What Python knowledge is actually required?	61
13.1.1 The smallest useful Python example	61
13.1.2 Conditions: turning observations into decisions	61
13.1.3 Functions: package your thinking	61
13.2 Reading an error instead of fearing it	62
13.3 EDA coding conventions from day one	62
14 EDA Laboratory Patterns You Should Reuse	63
14.1 The universal first-pass notebook	63
14.2 A reusable data-quality report function	63
14.2.1 Why this is better than a giant one-liner	64
14.3 Assertions: turn assumptions into executable checks	64
14.4 Groupby: one of the most important EDA ideas	64
14.5 Filtering: translate English into boolean logic	64
14.6 Vectorisation: think in columns	65
14.7 When a loop is actually useful	65
15 Visual Reasoning Workbook	66
15.1 Plot selection decision tree	66
15.2 Histogram: a measurement of your plotting choices	66
15.3 Box plot: understand the five-number story	66
15.4 Scatter plot: relationship, not proof of cause	67
15.5 ECDF: a powerful beginner-friendly distribution tool	67
16 Statistics Without Memorisation	68
16.1 Why different summaries answer different questions	68
16.2 Sample variance: derive it from deviation	68
16.3 IQR and robust spread	68
16.4 Skewness: reason about the tail	69
16.5 Kurtosis: avoid the “peakedness only” trap	69
16.6 Trimmed mean: a deliberate compromise	69
17 Missingness, Bias and Data Quality Reasoning	70
17.1 Missing is a state, not a number	70
17.2 A missingness investigation	70
17.3 MCAR, MAR and MNAR: a practical mental model	70
18 Cleaning Without Destroying Evidence	71
18.1 The cleaning decision tree	71
18.2 Duplicates: exact duplicate versus entity duplicate	71
18.3 Structural inconsistencies	71
19 Feature Engineering: From Raw Columns to Useful Signals	73
19.1 Feature engineering starts with a question	73
19.1.1 Interaction features	73
19.1.2 Polynomial features	73
19.1.3 Datetime features	73
19.1.4 Cyclical time	73
19.2 Feature-source ledger	74

20 Encoding and Scaling: Choosing Rather Than Memorising	75
20.1 Encoding decision table	75
20.2 Min-Max scaling	75
20.3 Z-score standardisation	75
20.4 L2 row normalisation	75
20.5 Robust scaling	76
21 Correlation, Covariance and Multivariate Thinking	77
21.1 Pearson correlation from standardised co-movement	77
21.2 Spearman and Kendall	77
21.3 Covariance matrix	77
21.4 VIF: why multicollinearity matters	77
22 PCA — From Geometry to Mathematics to Code	79
22.1 The two-dimensional picture	79
22.2 Eigenvectors and eigenvalues	79
22.3 Variance retention	79
22.4 PCA in code	80
23 Automated EDA and Dashboards — Use the Machine, Keep the Judgment	81
23.1 What automated EDA is good at	81
23.2 What automated EDA cannot decide reliably	81
23.3 A minimal Streamlit EDA dashboard	81
24 Domain Thinking Workbook	82
24.1 Healthcare	82
24.2 Retail	82
24.3 Customer cohorts	82
24.4 Finance	82
25 Job-Ready EDA: From Notebook to Portfolio	84
25.1 What an internship interviewer may actually ask	84
25.2 A portfolio-quality EDA repository	84
25.3 EDA report template	85
26 Exam Mastery and Logic-Building Workbook	86
26.1 How to write a strong theory answer	86
26.2 Example: “Explain outlier detection using IQR”	86
26.3 Code-to-logic translation drills	86
26.4 Debugging ladder	87
26.5 Find the bug — mixed types	87
26.6 Find the bug — chained assumptions	87
27 Independent Practice: Build Your Own Logic	88
27.1 The “no function name” challenge	88
27.2 The “predict the output” challenge	88
27.3 The “change one thing” challenge	88
28 Final Integrated Practical — The Student EDA Mission	90
28.1 Scenario	90
28.2 Mission rules	90
28.3 Required deliverables	90
28.4 Scoring rubric	91
28.5 Self-review before submission	91
29 Personal Improvement System for the Semester	92
29.1 The error journal	92

29.2 Weekly 30-minute improvement loop	92
Practical Dataset Ladder	93
References and Official Learning Resources	94
Closing Note	96

1

Unit 1 — Foundations of EDA & Environment Setup

1.1 The question behind EDA

Core Definition

Exploratory Data Analysis is the structured investigation of a dataset using summaries, visualisations, transformations and domain reasoning to understand its structure, quality, variability and relationships before important modelling or business decisions are made.

EDA answers four different questions that beginners often mix together: **What is present?** **How is it distributed?** **How do variables relate?** and **What could be misleading or wrong?**

Intuition & Real-World Analogy

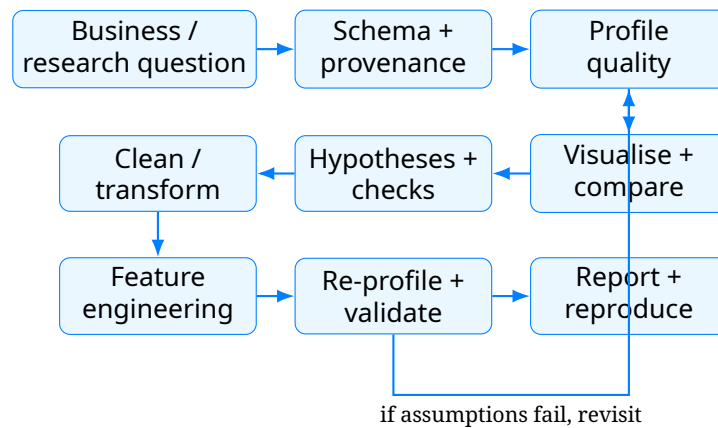
Imagine a mechanic receiving an unfamiliar car. A good mechanic does not immediately replace the engine. They inspect the vehicle, listen to sounds, check fluids, measure pressure, look for leaks, and only then choose a repair. EDA is that inspection stage for data. A model trained on an uninspected dataset is like repairing a car by guessing from its colour.

1.2 EDA versus data cleaning versus modelling

These activities overlap but are not identical. **EDA** seeks understanding; **cleaning** changes representation or removes invalid records; **feature engineering** constructs useful representations; **modelling** learns a predictive or descriptive function. EDA should influence cleaning, but cleaning decisions should remain traceable.

Activity	Main question	Typical output	Typical mistake
EDA	What is happening?	findings, plots, hypotheses	treating a plot as proof
Cleaning	What is invalid or unusable?	corrected dataset + audit log	deleting inconvenient observations
Feature engineering	How can information be represented?	derived variables	leaking future information
Modelling	Can we learn a useful function?	trained model + metrics	optimising before understanding data

1.3 The complete EDA lifecycle



1.3.1 Step 0 — Define the unit of observation

Before opening pandas, write one sentence: *“One row represents a single observational unit.”* Examples: one customer, one invoice line, one patient visit, one stock-day, one sensor reading. If this sentence is unclear, aggregation, duplicate detection and leakage analysis can all become wrong.

Common Pitfalls & Edge Cases

A dataset with 100,000 rows is not automatically a dataset with 100,000 independent observations. A retail transaction table may contain many rows per customer; a patient table may contain repeated visits; a time series contains serial dependence. Treating rows as independent without checking the data-generating process is a common analytical error.

1.3.2 Step 1 — Establish provenance

Record where the data came from, extraction date, schema version, units, filters, join keys, time window, sampling method, known exclusions and privacy constraints. A dataframe without provenance is difficult to trust six months later.

Hands-On Lab Pattern

Create a small data card before analysis. Minimum fields: source, row_grain, time_range, target, units, known_limitations, privacy_classification.

1.3.3 Step 2 — Profile structure before statistics

Check shape, columns, dtypes, unique counts, null counts, duplicate rows, constant columns, suspicious ranges and parsing failures. Statistics computed on a malformed column can be numerically correct and semantically useless.

```

1 import pandas as pd
2
3
4 def structural_audit(df: pd.DataFrame) -> pd.DataFrame:
5     return pd.DataFrame({
6         "dtype": df.dtypes.astype(str),
7         "missing_n": df.isna().sum(),
8         "missing_pct": df.isna().mean().mul(100).round(2),
9         "unique_n": df.nunique(dropna=False),
10        "memory_bytes": df.memory_usage(deep=True).iloc[1:],
11    }).sort_values("missing_pct", ascending=False)
12
13 print(df.shape)
14 print(df.head(3))
15 print(structural_audit(df))
  
```

```
16 print("Duplicate rows:", df.duplicated().sum())
```

1.3.4 Step 3 — Form hypotheses before hunting patterns

A plot can always be made to look interesting. A hypothesis gives the plot a purpose. Example: “Customers with a longer tenure may have lower churn.” The correct workflow is to define the variables, decide how to measure tenure/churn, inspect distributions, visualise the relationship, examine subgroup differences and ask whether alternative explanations exist.

Think Like an Engineer

Use the sentence pattern: **If X is associated with Y, then I expect to see Z in a comparison of A and B, after controlling for C where relevant.** This simple habit turns random plotting into analytical reasoning.

1.4 Data types: measurement scale, not just Python dtype

Python might store a variable as `int64`, `float64` or `object`. That is an implementation detail. Statistical meaning comes from the measurement scale.

1.4.1 Nominal

Categories are labels with no intrinsic order: city, browser, blood group, product family. Addition and subtraction are meaningless. Counts and proportions are meaningful.

1.4.2 Ordinal

Categories have order but unequal distances are not guaranteed: satisfaction = poor/fair/good/excellent, risk = low/medium/high. Median and rank-based methods can be meaningful; assuming equal spacing requires justification.

1.4.3 Interval

Differences are meaningful but zero is arbitrary. Temperature in Celsius is the classic example: 20°C is not “twice as hot” as 10°C. Means can be meaningful; ratios require care.

1.4.4 Ratio

A meaningful zero exists and ratios are interpretable: age, mass, revenue, duration. Most arithmetic operations have a natural interpretation.

1.4.5 Discrete versus continuous

Discrete variables count separated outcomes such as number of purchases. Continuous variables measure on a continuum such as height or latency. The distinction matters for visualisation and distribution assumptions, but real datasets often contain rounded continuous measurements.

Exam Thinking

Exam trap: “Integer” does not mean “discrete count” in the semantic sense, and “float” does not guarantee a continuous measurement. A ZIP code may be stored as an integer but is nominal. A rating from 1–5 is numeric-looking but often ordinal.

1.5 Structural layouts

1.5.1 Wide versus long

Wide data places repeated measures in multiple columns; long data uses rows to represent observations and a variable column to identify the measurement. Long format is often easier for grouped visualisation and aggregation.

```
1 wide = pd.DataFrame({
2     "student": ["A", "B"],
3     "math": [81, 74],
4     "python": [88, 91],
5 })
6
7 long = wide.melt(id_vars="student", var_name="subject", value_name="score")
8 print(long)
```

1.5.2 Tidy-data principles

A useful working rule is: each variable is a column, each observation is a row, and each observational unit forms a logical table. Violations are not automatically wrong, but they should be recognised before analysis.

1.6 Statistical distributions: learn the shape before the formula

A distribution describes how values are allocated across possible outcomes. Key ideas include centre, spread, skewness, tails, modality and dependence.

1.6.1 Common distribution shapes

- Symmetric and approximately bell-shaped: mean and median may be close.
- Right-skewed: many small/moderate values with a long upper tail; income and transaction amounts often behave this way.
- Left-skewed: many high values with a long lower tail.
- Heavy-tailed: extreme observations occur more often than a light-tailed reference would suggest.
- Bimodal: two subpopulations may be mixed; one global average can hide the structure.

Intuition & Real-World Analogy

A histogram is not merely a picture of “frequency.” It is evidence about the mechanism that may have generated the observations. Two groups merged into one column can create a strange-looking distribution; separating the groups may reveal two perfectly ordinary processes.

1.7 Python environment: reproducibility before exploration

1.7.1 Jupyter and Google Colab workflow

A notebook is both an interactive laboratory and a reproducible document. Keep cells ordered from environment -> imports -> data acquisition -> audit -> analysis -> conclusions. Restart and run all before sharing.

1.7.2 Dependency management

Pin major versions for reproducible projects and record the environment. Do not repeatedly install packages in random cells. For teaching notebooks, a first cell may install compatible packages; for production, use a lockfile or managed environment.

```

1 # Colab / notebook bootstrap
2 %pip install -q pandas numpy matplotlib seaborn scikit-learn scipy
3
4 import sys
5 import platform
6 import numpy as np
7 import pandas as pd
8 import matplotlib.pyplot as plt
9 import seaborn as sns
10
11 print("Python:", sys.version.split()[0])
12 print("Platform:", platform.platform())
13 print("pandas:", pd.__version__)
14 print("numpy:", np.__version__)
15
16 sns.set_theme(style="whitegrid", context="notebook")
17 plt.rcParams["figure.figsize"] = (9, 5)

```

1.8 Pandas fundamentals with reasoning

1.8.1 Series

A Series is a one-dimensional labelled array. Its index is part of its semantics, not decorative meta-data. When two Series interact, pandas aligns values by labels.

```

1 s = pd.Series([10, 20, 30], index=["a", "b", "c"], name="score")
2 t = pd.Series([100, 200], index=["c", "a"], name="bonus")
3 print(s + t)

```

Common Pitfalls & Edge Cases

The result above is label-aligned, not position-aligned. If you expected 10+100 and 20+200, you have silently written the wrong mental model. Pandas documentation explicitly treats index alignment as a core behaviour. [Read the current indexing guide](#).

1.8.2 DataFrame

A DataFrame is a two-dimensional labelled table where each column may have a different dtype. Think of it as a collection of aligned Series sharing a row index.

```

1 df = pd.DataFrame({
2     "age": [19, 21, 20],
3     "score": [82.0, 91.5, 77.0],
4     "city": ["Indore", "Bhopal", "Indore"],
5 })
6 print(df.info())
7 print(df.describe(numeric_only=True))

```

1.8.3 Indexing: label versus position

loc is primarily label-based; iloc is position-based. Explicit indexing makes intent visible and reduces subtle mistakes.

```

1 subset_a = df.loc[df["score"] >= 80, ["age", "score"]]
2 subset_b = df.iloc[:2, :2]

```

1.8.4 Vectorisation

Vectorisation expresses a whole-column operation rather than writing a Python loop over rows.

```
1 df["score_pct"] = df["score"] / 100
2 # Prefer this:
3 df["passed"] = df["score"] >= 40
4 # over manually iterating through rows.
```

Industry Practice

Vectorisation is not only about speed. It makes the transformation easier to read, test and reason about. For row-dependent logic that truly cannot be expressed vectorially, use carefully chosen tools such as `np.select`, `Series.map`, or `DataFrame.assign`; use `apply` only when it is the clearest correct option.

1.9 Unit 1 practical build: an EDA starter function

```
1 from typing import Any
2
3
4 def eda_snapshot(df: pd.DataFrame) -> dict[str, Any]:
5     """Return a compact, auditable first-pass profile."""
6     return {
7         "shape": df.shape,
8         "columns": df.columns.tolist(),
9         "dtypes": df.dtypes.astype(str).to_dict(),
10        "missing": df.isna().sum().to_dict(),
11        "duplicates": int(df.duplicated().sum()),
12        "numeric_summary": df.describe(numeric_only=True).to_dict(),
13    }
14
15 \section{Unit 1 dataset practice map}
16 Use small datasets first, then move to larger data. Suggested public resources include:\\
17 \dataset{https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic}{UCI Breast Cancer
18 Wisconsin Diagnostic};
19 \dataset{https://archive.ics.uci.edu/dataset/352/online+retail}{UCI Online Retail};
20 \dataset{https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce}{Brazilian E-Commerce Public Dataset
21 };
22 \dataset{https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data}{House Prices dataset};
23 \dataset{https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset}{Movies dataset}.
24 \begin{labbox}
25 For each dataset, do not begin with a model. Write: (1) row grain, (2) target or decision question, (3) 5
26 numerical columns, (4) 5 categorical columns, (5) likely missingness mechanism, (6) one possible
27 leakage path, and (7) one plot you expect to be informative.
28 \end{labbox}
29
30 \section{Unit 1 practice and debugging}
31 \subsection{Short-answer questions}
32 \begin{enumerate}[leftmargin=0]
33 \item Why can a dataset have correct numeric values but still be structurally wrong?
34 \item Explain why ZIP code should normally be treated as nominal.
35 \item What is the difference between an observation and a row?
36 \item Why should provenance be recorded before aggressive cleaning?
37 \item Explain pandas index alignment with a two-row example.
38 \end{enumerate}
39 \subsection{Find the bug}
40 \begin{lstlisting}
41 # BUG: student expects positional addition.
42 a = pd.Series([10, 20], index=["x", "y"])
43 b = pd.Series([100, 200], index=["y", "z"])
44 print(a + b)
```

Task: predict the output before running it. Then write two correct alternatives: one label-aligned and one position-based.

1.9.1 Logic-building lab

Given a table of students with attendance, assignment, and exam columns, define a reproducible rule for “needs support.” Your rule must state thresholds, missing-data behaviour and a reason for each condition. Implement it with vectorised pandas operations and test it on at least five hand-crafted edge cases.

1.10 Probability and distribution toolkit

EDA students do not need to become probability theorists before plotting data, but they do need enough probability intuition to recognise what a distribution means and why different summaries behave differently.

1.10.1 Population versus sample

A **population** is the full set of units relevant to a question. A **sample** is the subset actually observed. Population quantities are parameters; sample calculations are statistics used to learn about those parameters.

Intuition & Real-World Analogy

If a university wants the average sleep time of every enrolled student, the enrolled students are the population. If you survey 400 students, those 400 form the sample. The sample mean is not “the population mean”; it is an estimate of it, subject to sampling variability and bias.

1.10.2 Sampling variability

Two random samples from the same population will not generally have identical means. This is why a statistic should be interpreted with uncertainty rather than treated as a permanent property of reality.

1.10.3 Sampling bias

A large sample can still be biased. Common sources include convenience sampling, survivorship bias, non-response, selection by the outcome, and coverage gaps.

Think Like an Engineer

Ask: **Who had a chance to enter the dataset? Who did not?** This question often reveals more than another 20 plots.

1.10.4 Bernoulli distribution

A Bernoulli variable represents one binary trial with success probability p :

$$P(X = 1) = p, \quad P(X = 0) = 1 - p.$$

Its mean is p and variance is $p(1 - p)$. Examples include a single conversion, pass/fail outcome or default/non-default event.

1.10.5 Binomial distribution

If n independent Bernoulli trials share success probability p , the number of successes X follows

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}.$$

The assumptions matter: fixed number of trials, binary outcome, common probability and independence. Real customer events may violate independence because the same person can generate multiple observations.

```

1 from scipy.stats import binom
2
3 n = 100
4 p = 0.20
5 probb_exactly_25 = binom.pmf(25, n, p)
6 probb_at_most_25 = binom.cdf(25, n, p)
7 print(probb_exactly_25, probb_at_most_25)

```

1.10.6 Poisson distribution

The Poisson distribution models counts of events in a fixed interval under assumptions about a stable event rate and independent occurrence. Its probability mass function is

$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}.$$

Both mean and variance equal λ under the classical model.

Example questions: number of support tickets per hour, arrivals per minute, or defects per production batch. The model becomes questionable when rates change strongly over time or events cluster.

```

1 from scipy.stats import poisson
2
3 rate = 4
4 print("P(X=6):", poisson.pmf(6, rate))
5 print("P(X<=6):", poisson.cdf(6, rate))

```

1.10.7 Uniform distribution

A continuous uniform distribution gives equal density across an interval $[a, b]$:

$$f(x) = \frac{1}{b-a}, \quad a \leq x \leq b.$$

It is useful as a simple simulation mechanism, not as an assumption to impose on arbitrary real data.

1.10.8 Normal distribution

The normal density is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

It is symmetric around μ . The familiar 68–95–99.7 rule describes approximate mass within 1, 2 and 3 standard deviations for a normal distribution.

Common Pitfalls & Edge Cases

“Data should be normal” is not a general EDA rule. Many variables are naturally skewed, bounded, discrete or heavy-tailed. The right question is whether a model, statistical procedure or business decision requires a particular assumption.

1.10.9 Exponential distribution

The exponential distribution often models waiting times between events under a constant-rate Poisson process:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

It has a strong right tail and a useful memoryless property. Real service times may not satisfy that property, but the model is a helpful reference shape.

1.10.10 Student's t and chi-square distributions

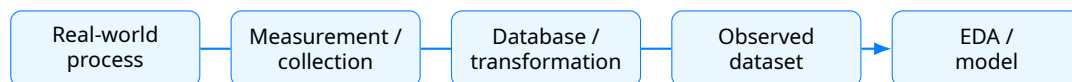
The Student's t distribution appears naturally when estimating a mean with unknown population standard deviation, especially for smaller samples. Its tails are heavier than the normal distribution. Chi-square distributions arise from sums of squared standard-normal quantities and are central to several variance and contingency-table procedures.

1.10.11 Why distribution knowledge helps EDA

Knowing reference distributions gives you vocabulary for questions such as: “Is this count roughly Poisson-like?”, “Does the tail look heavier than normal?”, “Is the variance changing over time?”, or “Could this binary rate plausibly come from a constant probability?” These are hypotheses, not automatic labels.

1.11 Data-generating process thinking

A dataset is the visible result of a process. For a retail transaction, the process may be: customer visits site -> product displayed -> customer chooses -> payment succeeds -> order recorded -> refund/cancellation may occur later. Each stage can introduce selection, missingness and leakage.



Errors can enter at every arrow. A missing value may be caused by a skipped measurement, a database failure, a policy rule or a participant decision. An outlier may be a typo or the legitimate output of an unusual process. EDA becomes much stronger when the analyst reasons about these mechanisms.

1.12 Measurement error and units

Never compare numerical columns without checking units. A temperature column may contain Celsius and Fahrenheit; weight may mix kilograms and pounds; revenue may be stored in rupees for one source and dollars for another.

```

1 # Example: explicit unit conversion
2 weight_kg = df["weight_lb"] * 0.45359237
3
4
5 \section{Sampling and time leakage}
6 Random train/test splitting is not automatically correct. For time-dependent prediction, future
   observations must not inform past predictions. For repeated entities, random row splitting can put
   the same customer into both training and test sets, producing overly optimistic evaluation.
7
8 \begin{lstlisting}
9 # Time-aware split idea
10 train = df[df["date"] < "2025-01-01"].copy()
11 test = df[df["date"] >= "2025-01-01"].copy()
  
```

Exam Thinking

A dataset can be large, clean and beautifully visualised while still being statistically invalid for the intended task. Always connect the split strategy to the data-generating process.

2

Unit 2 — Descriptive Statistics & Visual Discovery

2.1 The discipline of summarisation

A statistic is a compressed description of data. Compression is useful because humans cannot inspect every row, but compression loses information. The job of EDA is to choose summaries that preserve the signal relevant to the question.

Think Like an Engineer

Never ask only “What is the mean?” Ask: mean of what population, measured in what unit, over what period, after which exclusions, and is the mean representative of a typical observation?

2.2 Central tendency

2.2.1 Arithmetic mean

For observations $x_1, \dots, x_n$, the sample mean is

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

It uses every observation but is sensitive to extreme values.

Intuition & Real-World Analogy

If five students score 70, 72, 75, 76 and 77, the mean describes the centre well. If one value is accidentally entered as 700, the mean becomes a warning signal about data quality rather than a useful description of a typical student.

2.2.2 Median

The median is the middle ordered observation, or the average of the two middle observations for even n . It is robust to extreme values and is often preferable for skewed monetary or duration data.

2.2.3 Mode

The mode is the most frequent value. It is especially natural for categorical data and can be multi-modal.

2.2.4 Trimmed mean

A trimmed mean removes a specified proportion of observations from both tails before averaging. It offers a compromise between mean efficiency and median robustness. The trimming rule must be reported because different trimming levels produce different estimates.

```
1 from scipy.stats import trim_mean
2
3 x = np.array([10, 11, 11, 12, 12, 13, 14, 100])
4 print("mean:", x.mean())
5 print("median:", np.median(x))
6 print("trimmed:", trim_mean(x, proportiontocut=0.125))
```

2.3 Dispersion

2.3.1 Sample variance and standard deviation

The sample variance is

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2, \quad s = \sqrt{s^2}.$$

The $n-1$ denominator is the conventional unbiased estimator of population variance under standard assumptions. It is not a magic correction that makes every dataset unbiased.

2.3.2 IQR

$$IQR = Q_3 - Q_1.$$

The common Tukey fences are

$$L = Q_1 - 1.5IQR, \quad U = Q_3 + 1.5IQR.$$

These are rules for flagging unusual observations, not universal laws of nature.

2.3.3 MAD

Median absolute deviation is based on deviations from the median:

$$MAD = \text{median}(|x_i - \text{median}(x)|).$$

A normal-consistency scaling factor is often approximately 1.4826, but use it only when the interpretation calls for that convention.

2.3.4 Coefficient of variation

For positive measurements where a ratio scale is meaningful,

$$CV = \frac{s}{\bar{x}}.$$

It is not meaningful when the mean is near zero or when the zero point is arbitrary.

2.4 Skewness and kurtosis

Skewness describes asymmetry; kurtosis is related to tail weight and concentration relative to a reference. Different software conventions exist, so a reported number should identify the estimator or library convention when precision matters.

Common Pitfalls & Edge Cases

Do not say “high kurtosis means the distribution is peaked” and stop there. In practical EDA, tail behaviour is often more important than the visual height of the centre. Inspect the tails directly with quantiles, box plots or log-scaled views.

2.5 Visual discovery

2.5.1 Histogram

A histogram partitions values into bins. Bin width changes the apparent story, so inspect more than one sensible binning when the conclusion depends on shape.

```
1 sns.histplot(data=df, x="income", bins=30, kde=True)
2 plt.xlabel("Monthly income")
3 plt.ylabel("Count")
4 plt.title("Income distribution")
5 plt.show()
```

2.5.2 KDE

Kernel density estimation constructs a smooth estimate of a distribution. The bandwidth controls smoothness: too small can create noise; too large can erase meaningful modes.

2.5.3 Box plot

A box plot shows median, quartiles and potential outliers according to its whisker rule. It is excellent for comparing distributions across groups.

```
1 sns.boxplot(data=df, x="department", y="salary")
2 plt.xticks(rotation=30)
3 plt.show()
```

2.5.4 Violin plot

A violin plot combines distribution shape with summary information. It is useful when two groups have the same median but different density patterns.

2.5.5 Scatter plot

Scatter plots reveal direction, strength, non-linearity, clusters and unusual observations in two numeric variables.

```
1 sns.scatterplot(data=df, x="experience", y="salary", hue="role", alpha=0.65)
2 plt.title("Experience versus salary by role")
3 plt.show()
```

2.5.6 Pair plot

Pair plots provide a compact multivariate screening view, but they become unreadable as the number of variables grows. Use them to generate hypotheses, not to replace targeted analysis.

```
1 cols = ["age", "income", "spend", "tenure", "churn"]
2 sns.pairplot(df[cols], hue="churn", corner=True)
3 plt.show()
```

2.6 Univariate, bivariate and multivariate reasoning

Level	Question	Typical tools	Failure mode
Univariate	What does one variable look like?	histogram, KDE, box plot, quantiles	ignoring subgroups
Bivariate	How do two variables relate?	scatter, grouped box, contingency table	confusing association with cause
Multivariate	Does the relationship change with context?	pair plots, correlation matrix, stratification	Simpson's paradox / confounding

2.7 Outlier detection

2.7.1 Z-score

$$z_i = \frac{x_i - \bar{x}}{s}.$$

A threshold such as $|z| > 3$ is a screening rule, not a universal deletion rule. It works poorly when distributions are strongly skewed or heavy-tailed.

```
1 from scipy.stats import zscore
2
3 z = zscore(df["income"], nan_policy="omit")
4 outlier_mask = np.abs(z) > 3
```

2.7.2 IQR rule

```
1 q1 = df["income"].quantile(0.25)
2 q3 = df["income"].quantile(0.75)
3 iqr = q3 - q1
4 mask = df["income"].between(q1 - 1.5 * iqr, q3 + 1.5 * iqr)
```

Think Like an Engineer

When you find an outlier, ask four questions in order: **Is it impossible?** **Is it a data-entry error?** **Is it rare but valid?** **Does the business process produce legitimate extremes?** Only the first two strongly support correction/removal without further modelling.

2.8 Missingness profiling

Missingness is itself data. Start with counts, percentages, row patterns and relationships with observed variables.

```
1 missing = (
2     df.isna()
3     .mean()
4     .mul(100)
5     .sort_values(ascending=False)
```

```
6         .rename("missing_pct")
7         .to_frame()
8     )
9     print(missing)
10
11     sns.heatmap(df.isna(), cbar=False)
12     plt.title("Missingness pattern")
13     plt.show()
```

2.9 Worked reasoning example: the mean lies

Suppose delivery times are [25, 26, 27, 28, 29, 120]. Mean = 42.5, median = 27.5. If the question is “What does a normal customer experience?”, the median may be more representative. If the question is “What total staffing load should we expect per job?”, the extreme 120-minute case may be operationally important and should not simply be deleted.

2.10 Unit 2 practical lab

Hands-On Lab Pattern

Generate a synthetic customer-spend dataset with 1,000 rows. Create a log-normal spend distribution, inject 10 extreme values, inject missing values into two columns with different mechanisms, and create a segment variable. Your task: compare mean/median, identify outliers using both methods, visualise the distribution before and after log transformation, and write five evidence-based findings.

2.11 Unit 2 exercises

2.11.1 MCQs and short answers

1. Which measure is usually more robust to a single extreme value: mean or median? Explain.
2. Why can a KDE hide a small subgroup?
3. When can a Z-score be misleading?
4. Why is an outlier flag not automatically a deletion rule?
5. Explain why missingness percentages should be computed before imputation.

2.11.2 Find the bug

```
1 # BUG: this overwrites the entire DataFrame with a boolean Series.
2 df = df["income"] > 100000
```

Repair the code while preserving the original dataframe and producing a filtered view.

3

Unit 3 — Data Cleaning & Feature Engineering

3.1 Cleaning is a decision system

Cleaning is not “making the dataframe pretty.” It is deciding which representation is faithful enough for the analytical question. Every transformation should answer: *What problem does this solve, what information might it destroy, and how will I verify the result?*

3.2 Missing data: MCAR, MAR, MNAR

3.2.1 MCAR — Missing Completely At Random

Missingness is unrelated to observed or unobserved values. This is a strong assumption and rarely established by simply looking at a heatmap.

3.2.2 MAR — Missing At Random

Missingness depends on observed variables. For example, income may be missing more often for a particular age group after conditioning on age and employment status.

3.2.3 MNAR — Missing Not At Random

Missingness depends on the unobserved value itself or another unobserved factor. For example, people with unusually high debt may be less willing to report debt.

Intuition & Real-World Analogy

Imagine a survey asking income. If ten people forget randomly, that is different from people with high income systematically refusing to answer. Filling both cases with the median may produce the same code but very different statistical consequences.

3.2.4 A practical missingness investigation

1. Measure missingness by column.
2. Measure missingness by row.
3. Compare missingness across important groups.

4. Check whether missingness is associated with the target.
5. Determine whether the value has a meaningful “not applicable” state.
6. Decide whether to drop, impute, add an indicator, or redesign collection.

```

1 df["income_missing"] = df["income"].isna().astype("int8")
2 missing_by_group = (
3     df.groupby("employment_status", dropna=False)["income_missing"]
4         .mean()
5         .mul(100)
6         .round(2)
7 )
8 print(missing_by_group)

```

3.2.5 Imputation strategies

Mean, median, mode, constant values, forward/backward fill, group-wise imputation, KNN and model-based methods each encode assumptions. Median imputation is robust but can shrink variance; forward fill may be sensible in some time-series contexts but dangerous across long gaps.

Common Pitfalls & Edge Cases

Never fit an imputer on the full dataset before train/test splitting when the resulting values influence a model evaluation. The imputation statistic itself can leak information from the test set. Use a pipeline so fitting happens on training data only.

3.3 Deduplication

There are at least three meanings of “duplicate”: identical rows, repeated business events, and repeated entities represented with different spellings. Use a business key whenever possible.

```

1 exact_dupes = df.duplicated(keep=False)
2 print(df.loc[exact_dupes].sort_values(df.columns.tolist()))
3
4 # Example business key
5 business_dupes = df.duplicated(subset=["customer_id", "invoice_id"], keep=False)

```

3.4 Structural inconsistencies

Typical issues include inconsistent casing, whitespace, spelling variants, impossible dates, mixed units, placeholder values such as -999, and category drift.

```

1 df["city"] = (
2     df["city"]
3     .astype("string")
4     .str.strip()
5     .str.casefold()
6 )
7
8 city_map = {
9     "indore city": "indore",
10    "indore ": "indore",
11 }
12 df["city"] = df["city"].replace(city_map)
13
14 \section{Type casting and semantic parsing}
15 \subsection{Numeric conversion}
16 \begin{lstlisting}
17 df["amount"] = pd.to_numeric(df["amount"], errors="coerce")

```

3.4.1 Datetime conversion

```
1 df["order_date"] = pd.to_datetime(
2     df["order_date"], errors="coerce", utc=True
3 )
```

3.4.2 Categoricals

Use categorical dtype when a column has repeated finite categories and the memory/performance or semantic benefits justify it.

3.5 Scaling and transformation

3.5.1 Min-Max scaling

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}.$$

It maps training extrema to 0 and 1 but is sensitive to future values outside the observed range.

3.5.2 L2 normalization

For a vector x ,

$$\hat{x} = \frac{x}{\|x\|_2}, \quad \|x\|_2 = \sqrt{\sum_j x_j^2}.$$

This normalises each sample vector, which is conceptually different from scaling each feature across samples.

3.5.3 Z-score standardisation

$$z = \frac{x - \mu}{\sigma}.$$

This is useful for algorithms sensitive to feature scale, but it does not magically make a non-normal distribution normal.

3.5.4 Robust scaling

Robust scaling uses median and IQR, making it less sensitive to extreme values. Scikit-learn documents robust scaling as a useful alternative when outliers are present. [Current preprocessing guide](#).

3.5.5 Log and power transforms

For positive skewed variables, $\log(1 + x)$ is common when zeros exist. Power transforms such as Yeo-Johnson extend the idea to values that may be zero or negative.

```
1 from sklearn.preprocessing import PowerTransformer
2
3 pt = PowerTransformer(method="yeo-johnson", standardize=True)
4 X_transformed = pt.fit_transform(X_train)
5 X_test_transformed = pt.transform(X_test)
```

3.5.6 Discretisation / binning

Binning can make rules easier to explain but loses within-bin information. Quantile bins create roughly equal observation counts; fixed-width bins preserve domain intervals.

```
1 df["age_band"] = pd.cut(
2     df["age"],
3     bins=[0, 18, 30, 45, 60, np.inf],
4     labels=["minor", "young_adult", "adult", "senior", "older"],
5     right=False,
6 )
```

3.6 Categorical encoding

3.6.1 One-hot encoding

Each category receives a binary indicator. It is appropriate for nominal variables when dimensionality is manageable.

3.6.2 Ordinal encoding

Maps ordered categories to numbers. The mapping must reflect a real order; arbitrary numeric labels can create false geometry.

3.6.3 Frequency encoding

Replace each category by its observed frequency. It is compact but can merge categories with similar frequencies and can encode dataset-specific information.

3.6.4 Target encoding

Replace a category by a target statistic such as mean target. This is powerful but extremely leakage-prone. It must be fitted within training folds, with smoothing and out-of-fold logic where appropriate.

```
1 from sklearn.compose import ColumnTransformer
2 from sklearn.preprocessing import OneHotEncoder, StandardScaler
3 from sklearn.pipeline import Pipeline
4 from sklearn.impute import SimpleImputer
5
6 numeric_pipe = Pipeline([
7     ("imputer", SimpleImputer(strategy="median")),
8     ("scale", StandardScaler()),
9 ])
10
11 categorical_pipe = Pipeline([
12     ("imputer", SimpleImputer(strategy="most_frequent")),
13     ("onehot", OneHotEncoder(handle_unknown="ignore")),
14 ])
15
16 preprocessor = ColumnTransformer([
17     ("num", numeric_pipe, numeric_cols),
18     ("cat", categorical_pipe, categorical_cols),
19 ])
```

3.7 Feature engineering

3.7.1 Interaction features

An interaction captures the possibility that the effect of one variable depends on another. For example, $price \times quantity$ gives revenue at line-item level.

3.7.2 Polynomial features

For features x_1, x_2 , a second-degree expansion can include x_1^2, x_2^2, x_1x_2 . This increases feature count rapidly and should be justified by a model or domain hypothesis.

```
1 from sklearn.preprocessing import PolynomialFeatures
2
3 poly = PolynomialFeatures(degree=2, include_bias=False)
4 X_poly = poly.fit_transform(X_train)
```

3.7.3 Datetime parsing

A timestamp is a container of many potential variables: year, month, weekday, hour, elapsed time, holiday status, recency, tenure and cyclical position.

```
1 dt = pd.to_datetime(df["timestamp"], errors="coerce")
2 df["year"] = dt.dt.year
3 df["month"] = dt.dt.month
4 df["weekday"] = dt.dt.dayofweek
5 df["hour"] = dt.dt.hour
6
7 \subsection{Cyclical time encoding}
8 For periodic variables such as hour-of-day,
9 \[
10 sin(2\pi h/24), \cos(2\pi h/24)
11 \]
12 can represent the wrap-around between 23:00 and 00:00 more naturally than a raw integer.
13
14 \section{Feature selection}
15 \subsection{Variance threshold}
16 A near-constant feature often contributes little information. However, low variance does not imply low
17 predictive value when a rare event is important.
18
19 \begin{lstlisting}
20 from sklearn.feature_selection import VarianceThreshold
21
22 selector = VarianceThreshold(threshold=0.01)
23 X_reduced = selector.fit_transform(X_train)
```

3.7.4 Filter-based metrics

Common filters include correlation with a target, mutual information, chi-square for appropriate categorical/count settings and univariate statistical tests. Filters are fast but may miss interactions.

3.8 Leakage: the hidden failure mode

Data leakage occurs when information unavailable at prediction time influences training or feature construction. Leakage can happen through future timestamps, target-derived features, group duplication across train/test, preprocessing fitted globally, and target encoding performed before splitting.

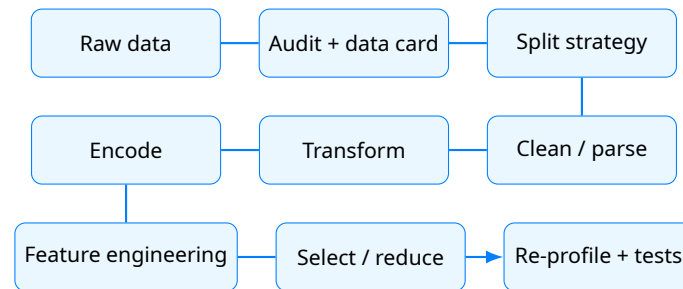
Intuition & Real-World Analogy

Imagine predicting whether a customer will cancel on Friday. A feature called `final_refund_amount` may predict cancellation extremely well—but only because the refund happens after the cancellation decision. The model is not “smart”; the feature is a time machine.

```
1 # Safe pattern: split first, fit transformations on training data only.
2 from sklearn.model_selection import train_test_split
3
4 X_train, X_test, y_train, y_test = train_test_split(
5     X, y, test_size=0.2, random_state=42, stratify=y
6 )
7 preprocessor.fit(X_train)
```

```
8 X_train_t = preprocessor.transform(X_train)
9 X_test_t = preprocessor.transform(X_test)
```

3.9 Unit 3 integrated cleaning workflow



3.10 Unit 3 lab: build a cleaning contract

For a messy dataset, create a table with columns: issue, evidence, rule, action, risk, verification. This turns “cleaning” into an auditable engineering process.

3.11 Unit 3 exercises

1. Design an imputation strategy for a skewed salary variable and defend it.
2. Give an example where deleting duplicates would delete valid repeated events.
3. Explain why target encoding can leak.
4. Construct a leakage example using a datetime column.
5. Compare StandardScaler and RobustScaler on a synthetic dataset with five extreme values.

4

Unit 4 — Multivariate Analysis, PCA & Reporting

4.1 Correlation is a lens, not a verdict

A correlation coefficient compresses a relationship into one number. The number cannot tell you whether the relationship is causal, nonlinear, driven by one subgroup, or created by a common confounder.

4.2 Pearson correlation

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}.$$

Pearson measures linear association. It is sensitive to outliers and can miss strong nonlinear relationships.

4.3 Spearman correlation

Spearman correlation is Pearson correlation computed on ranks. It captures monotonic relationships even when the relationship is not linear.

4.4 Kendall Tau

Kendall's tau is based on concordant and discordant pairs and is often useful when sample sizes are modest or ties/ranks matter. It has a direct interpretation in terms of pair ordering.

```
1 from scipy.stats import pearsonr, spearmanr, kendalltau
2
3 r_p, p_p = pearsonr(df["x"], df["y"])
4 r_s, p_s = spearmanr(df["x"], df["y"])
5 r_k, p_k = kendalltau(df["x"], df["y"])
```

4.5 Covariance matrix

For centered data matrix $X \in \mathbb{R}^{n \times p}$,

$$\Sigma = \frac{1}{n-1} X^T X.$$

Diagonal entries are feature variances; off-diagonal entries are pairwise covariances.

```

1 X = df[numeric_cols]
2 X_centered = X - X.mean()
3 cov = X_centered.cov()
4 cor = X.corr()
5
6 sns.heatmap(cor, annot=True, fmt=".2f", cmap="vlag", center=0)
7 plt.title("Correlation matrix")
8 plt.show()

```

4.6 Multicollinearity and VIF

Variance Inflation Factor for predictor j is

$$VIF_j = \frac{1}{1 - R_j^2},$$

where R_j^2 is obtained by regressing predictor j on the remaining predictors. High VIF indicates that the predictor contains information strongly overlapping with other predictors, making coefficient interpretation unstable in models such as ordinary linear regression.

```

1 from statsmodels.stats.outliers_influence import variance_inflation_factor
2
3 X_vif = df[numeric_cols].dropna().assign(const=1.0)
4 vif = pd.DataFrame({
5     "feature": X_vif.columns,
6     "VIF": [
7         variance_inflation_factor(X_vif.values, i)
8         for i in range(X_vif.shape[1])
9     ],
10 })
11 print(vif)

```

Common Pitfalls & Edge Cases

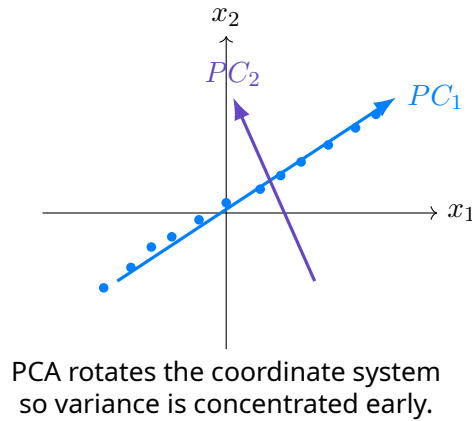
VIF is not a universal “bad feature detector.” A high VIF may be expected when variables intentionally encode the same construct. The correct response depends on the modelling goal: prediction may tolerate redundancy while coefficient interpretation may not.

4.7 PCA: principal component analysis

PCA finds orthogonal directions that capture decreasing amounts of variance. It is a dimensionality-reduction method, not a feature-importance method and not a causal discovery method.

4.7.1 Geometric intuition

Imagine a cloud of points in two dimensions shaped like a long diagonal ellipse. PCA rotates the coordinate system so that the first axis points along the longest direction of the cloud. The second axis is perpendicular and captures the remaining variation.



4.7.2 Step 1 — Center the features

For each feature j ,

$$X_{ij}^c = X_{ij} - \bar{X}_j.$$

Centering is essential because PCA otherwise depends on the coordinate origin.

4.7.3 Step 2 — Form covariance

$$\Sigma = \frac{1}{n-1} (X^c)^T X^c.$$

If feature scales differ greatly, standardisation may be appropriate before PCA.

4.7.4 Step 3 — Solve the eigenproblem

Find vectors v and values λ satisfying

$$\Sigma v = \lambda v.$$

Eigenvectors define principal directions; eigenvalues quantify variance along those directions.

4.7.5 Step 4 — Rank components

Sort eigenvalues from largest to smallest. The explained-variance ratio for component k is

$$EV R_k = \frac{\lambda_k}{\sum_j \lambda_j}.$$

The cumulative sum tells how much variance is retained by the first k components.

```

1 from sklearn.preprocessing import StandardScaler
2 from sklearn.decomposition import PCA
3
4 scaler = StandardScaler()
5 X_scaled = scaler.fit_transform(X_train)
6
7 pca = PCA()
8 X_pca = pca.fit_transform(X_scaled)
9
10 explained = pca.explained_variance_ratio_
11 cumulative = np.cumsum(explained)
12
13 plt.plot(range(1, len(explained) + 1), cumulative, marker="o")
14 plt.axhline(0.95, linestyle="--", label="95% variance")
15 plt.xlabel("Number of components")
16 plt.ylabel("Cumulative explained variance")
17 plt.legend()
18 plt.show()

```

Common Pitfalls & Edge Cases

Do not say “PCA preserves 95

4.8 Scree plots and component selection

A scree plot displays eigenvalues or explained variance by component. Look for an elbow, cumulative variance targets and downstream validation. The final number should be justified by the use case rather than selected mechanically.

4.9 Automated EDA reports

Tools such as ydata-profiling can rapidly generate statistics, visualisations and data-quality alerts. This is useful for triage, not a substitute for human interpretation. Current documentation describes automated univariate profiling, missingness, duplicates and relationship summaries. [YData Profiling documentation](#).

```
1 from ydata_profiling import ProfileReport
2
3 profile = ProfileReport(df, title="EDA Profiling Report", explorative=True)
4 profile.to_file("eda_profile.html")
```

4.10 Streamlit dashboard design

A useful EDA dashboard is not a notebook pasted into a webpage. It should answer a small set of user questions: What is the dataset health? Which segments differ? What changed over time? Where are anomalies? Which filters alter the conclusion?

```
1 import streamlit as st
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5
6 st.set_page_config(page_title="EDA Explorer", layout="wide")
7 st.title("EDA Explorer")
8
9 uploaded = st.file_uploader("Upload CSV", type=["csv"])
10 if uploaded is not None:
11     df = pd.read_csv(uploaded)
12     st.metric("Rows", f"{len(df):,}")
13     st.metric("Columns", f"{df.shape[1]:,}")
14     st.dataframe(df.head(20), use_container_width=True)
15
16     numeric = df.select_dtypes(include="number").columns.tolist()
17     if numeric:
18         feature = st.selectbox("Choose a numeric feature", numeric)
19         fig, ax = plt.subplots()
20         sns.histplot(df[feature].dropna(), kde=True, ax=ax)
21         ax.set_title(f"Distribution of {feature}")
22         st.pyplot(fig)
```

4.11 Unit 4 exercises

1. Construct a dataset where Pearson correlation is near zero but a strong nonlinear relationship exists. Explain the result.
2. Explain why correlation does not imply causation using a confounding example.
3. Compute a covariance matrix by hand for a tiny two-feature dataset.

4. Explain VIF conceptually without using software.
5. Perform PCA twice: once on raw features and once after standardisation. Explain the difference.
6. Build a Streamlit dashboard that lets the user select a feature and subgroup.

5

Unit 5 — Domain Case Studies & Production Capstone

5.1 Why domain context changes EDA

The same statistic can have different meanings in different domains. In healthcare, a rare event may be clinically important; in retail, repeated customers create hierarchical observations; in finance, time dependence and volatility invalidate many naive assumptions.

5.2 Healthcare EDA

5.2.1 Class imbalance

A dataset with 98% negatives and 2% positives can produce 98% accuracy by predicting the majority class every time. EDA should inspect class counts, subgroup rates, confusion-matrix-relevant distributions and sampling design before model selection.

```
1 class_rate = df["outcome"].value_counts(normalize=True).mul(100)
2 print(class_rate)
3
4 sns.countplot(data=df, x="outcome")
5 plt.title("Outcome distribution")
6 plt.show()
```

5.2.2 Privacy and HIPAA-aware handling

For educational work, prefer de-identified or synthetic data. Avoid unnecessary names, addresses, phone numbers, direct identifiers and free-text fields that can contain identifying information. Access controls, logging, retention and minimum-necessary principles matter when handling real protected health information.

Industry Practice

The technical lesson is broader than healthcare: **data access is part of data engineering**. A perfectly written EDA notebook can still be unacceptable if it copies sensitive data into uncontrolled locations, logs identifiers, or exports raw rows into reports.

5.3 Retail EDA

5.3.1 Time seasonality

Retail demand often changes by weekday, month, festival period, promotion and store. Always distinguish trend from seasonality and calendar effects.

```

1 df["order_date"] = pd.to_datetime(df["order_date"])
2 daily = (
3     df.set_index("order_date")["revenue"]
4     .resample("D")
5     .sum()
6 )
7
8 daily.plot(title="Daily revenue")
9 plt.ylabel("Revenue")
10 plt.show()

```

5.3.2 Customer cohorts

A cohort groups customers by a meaningful starting event such as first purchase month. Cohort retention compares customers who began at different times without mixing them into one aggregate lifetime curve.

```

1 df["order_month"] = df["order_date"].dt.to_period("M")
2 first_purchase = df.groupby("customer_id")["order_month"].transform("min")
3 df["cohort_month"] = first_purchase
4
5 df["months_since_first"] = (
6     (df["order_month"].dt.year - df["cohort_month"].dt.year) * 12
7     + (df["order_month"].dt.month - df["cohort_month"].dt.month)
8 )

```

5.4 Finance EDA

5.4.1 Returns

For price P_t , simple return is

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}},$$

while log return is

$$g_t = \ln(P_t) - \ln(P_{t-1}).$$

Returns are often more useful than raw price levels for analysing movement, but the correct representation depends on the question.

5.4.2 Volatility

A rolling standard deviation can provide a local measure of variability:

$$\sigma_t = Std(r_{t-w+1}, \dots, r_t).$$

The window w changes the time scale of the measurement.

```

1 df["return"] = df["close"].pct_change()
2 df["rolling_vol"] = df["return"].rolling(20).std()
3
4 \subsection{Non-stationarity}
5 A time series is non-stationary when properties such as mean, variance or dependence change over time. A
   global mean can therefore be misleading. Always plot the series, rolling statistics and relevant
   subperiods before applying assumptions developed for stationary processes.
6
7 \section{Causal thinking without pretending EDA proves causality}
8 EDA can reveal temporal ordering, confounding patterns, selection effects and subgroup differences. It
   can help generate causal hypotheses, but observational plots alone rarely establish causality.
9
10 \begin{center}

```

```

11 \begin{tikzpicture}[node distance=8mm and 9mm, every node/.style={font=\sffamily\small}, n/.style={circle,
12   draw=aurblue,fill=glacier,minimum size=1cm}, a/.style={-Latex,thick,draw=aurblue}]
13 \node[n](c){C};\node[n,right=of c](x){X};\node[n,right=of x](y){Y};\draw[a](c)--(x);\draw[a](c)--(y);\
14   draw[a](x)--(y);
15 \node[below=8mm of x,align=center,text width=9cm,font=\sffamily\scriptsize]{If C influences both X and Y,
16   the observed X--Y relationship can mix direct association with confounding. EDA should prompt
17   subgroup and design questions, not a causal claim.};
18 \end{tikzpicture}
19 \end{center}
20
21 \section{Production-grade capstone: the complete EDA system}
22 The capstone should not be “make 10 plots”. It should resemble an engineering project with a reproducible
23   input, clear question, audit trail and final decision memo.
24
25 \subsection{Capstone architecture}
26 \begin{center}
27 \begin{tikzpicture}[node distance=7mm and 7mm, every node/.style={font=\sffamily\scriptsize}, b/.style={
28   rounded corners,draw=aurblue,fill=glacier,minimum width=2.7cm,minimum height=10mm,align=center}, a/.
29   style={-Latex,thick,draw=aurblue}]
30 \node[b](source){Source / dataset};
31 \node[b,right=of source](ingest){Ingestion +\provenance};
32 \node[b,right=of ingest](raw){Raw snapshot};
33 \node[b,below=of raw](audit){Schema +\quality audit};
34 \node[b,left=of audit](eda){EDA +\hypotheses};
35 \node[b,left=of eda](clean){Cleaning\contract};
36 \node[b,below=of clean](feat){Feature\engineering};
37 \node[b,right=of feat](report){Report +\dashboards};
38 \node[b,right=of report](decision){Decision /\next experiment};
39 \draw[a](source)--(ingest)--(raw)--(audit)--(eda)--(clean)--(feat)--(report)--(decision);
40 \end{tikzpicture}
41 \end{center}
42
43 \subsection{Capstone Phase 1 – Problem framing}
44 Write a one-page problem statement: stakeholder, decision, unit of observation, target, prediction/
45   analysis horizon, acceptable errors, constraints and what success means. A vague problem creates a
46   vague EDA.
47
48 \subsection{Phase 2 – Data contract}
49 Define columns, types, units, valid ranges, null policy, key constraints, expected cardinalities and
50   temporal boundaries. Add automated assertions.
51
52 \begin{lstlisting}
53 def validate_contract(df: pd.DataFrame) -> None:
54     required = {"customer_id", "order_date", "amount"}
55     missing = required.difference(df.columns)
56     assert not missing, f"Missing required columns: {missing}"
57     assert df["customer_id"].notna().all(), "customer_id contains nulls"
58     assert (df["amount"] >= 0).all(), "Negative amount found"

```

5.4.3 Phase 3 — Reproducible profiling

Save raw snapshots, environment information, profiling tables and analysis outputs. Never overwrite the raw input during exploration.

5.4.4 Phase 4 — Distribution and relationship investigation

For every important numeric feature, inspect quantiles, histogram/KDE, box plot and missingness. For categorical features, inspect frequency, rare categories and target rates where appropriate. For relationships, use scatter/grouped summaries and stratify important confounders.

5.4.5 Phase 5 — Cleaning contract

Every change should be written as a rule: “If condition, then action, because reason.” Record row counts before and after each operation.

```

1 log = []
2
3 def log_step(name: str, before: int, after: int, reason: str) -> None:
4     log.append({
5         "step": name,
6         "before_rows": before,

```

```

7     "after_rows": after,
8     "rows_removed": before - after,
9     "reason": reason,
10  })

```

5.4.6 Phase 6 — Feature engineering

Engineer only after understanding the raw variables. For each feature record source columns, formula, time availability and leakage risk.

5.4.7 Phase 7 — Validation

Re-run structural audit, missingness, ranges, duplicate checks and key distributions. Compare before/after summaries. A transformation is incomplete until its effect is measured.

5.4.8 Phase 8 — Final report

A professional report should contain: executive summary, data description, quality findings, major distributions, relationships, cleaning decisions, feature decisions, limitations, unanswered questions and recommended next actions.

5.5 Capstone dataset ladder

1. **Beginner:** Palmer Penguins — small, visual, ideal for distributions and relationships. [Dataset site](#).
2. **Intermediate:** UCI Online Retail — transactions, customers, quantities, dates. [UCI dataset page](#).
3. **Advanced:** Brazilian E-Commerce Public Dataset — multiple related tables, dates, categories and customer/order structure. [Dataset page](#).
4. **Domain option:** UCI Breast Cancer Wisconsin Diagnostic — clinical-style classification structure. [UCI dataset page](#).

5.6 Capstone grading rubric

Dimension	Evidence expected	Weight
Problem framing	clear decision, unit, target, horizon	10%
Data understanding	schema, provenance, grain, quality audit	15%
Statistical reasoning	appropriate summaries and interpretation	15%
Visual reasoning	targeted plots with written findings	15%
Cleaning	justified, reproducible transformations	15%
Feature engineering	useful, non-leaky features	10%
Validation	before/after checks and limitations	10%
Communication	readable report/dashboard	10%

5.7 Independent thinking drills

These exercises deliberately remove the exact command you should use.

1. You receive a CSV with 30 columns and no documentation. Write your first 15 questions before writing code.
2. Two groups have the same average salary but visibly different distributions. Design the analysis that explains why.

3. A feature has 40% missingness and missingness is strongly associated with the target. Give three possible explanations and three different actions.
4. PCA reduces 20 features to 5 components and retains 92% variance. Give two reasons why a predictive model might still perform worse.
5. A dashboard shows a strong relationship between advertising spend and sales. List five reasons this may not mean advertising caused the increase.

6

Cross-Cutting Engineering Skills

6.1 How to read a dataset like an engineer

Use the five-layer inspection model:

1. **Semantics:** what does each field mean?
2. **Structure:** how are rows and keys organised?
3. **Statistics:** what are the distributions and relationships?
4. **Process:** how could the data have been generated?
5. **Decision:** what action will this analysis support?

6.2 The evidence ladder

A useful mental model is to move from weak evidence to stronger evidence:

single value → summary → distribution → comparison
→ stratified comparison → designed test/experiment

The farther you move to the right, the more carefully you must specify assumptions and design.

6.3 Reproducibility checklist

- Raw data is preserved and identified by version/date.
- Environment/dependencies are recorded.
- Random seeds are fixed where randomness affects results.
- Transformations are explicit and ordered.
- Train/test boundaries are respected.
- Figures have labels, units and titles.
- Every important conclusion has a traceable calculation or visual.
- Limitations are written next to conclusions rather than hidden at the end.

6.4 Debugging methodology for EDA code

When a result looks wrong, do not immediately rewrite the whole notebook. Reduce the problem.

1. Print shape and dtypes.
2. Inspect 5–10 rows around the suspicious case.
3. Check nulls and unique values.
4. Reproduce the calculation on a tiny hand-built dataset.
5. Compare expected and actual intermediate values.
6. Only then modify the implementation.

Hands-On Lab Pattern

Micro-debug exercise: Build a dataframe with indices [10, 20, 30]. Build a Series with indices [30, 10]. Predict the result of addition, assignment and concatenation. Then run it and explain every output row. This one exercise teaches more about pandas semantics than memorising ten indexing methods.

6.5 Exam strategy: answer the reasoning, not just the definition

For a theory question, use the structure: **definition** -> **formula/algorithm** -> **interpretation** -> **example** -> **limitation**. For a coding question: **input assumptions** -> **operation** -> **output** -> **validation**. For debugging: **symptom** -> **hypothesis** -> **minimal test** -> **fix** -> **regression check**.

6.6 Master formula sheet

$$\begin{aligned} \bar{x} &= \frac{1}{n} \sum x_i & s^2 &= \frac{1}{n-1} \sum (x_i - \bar{x})^2 & IQR &= Q_3 - Q_1 \\ z_i &= \frac{x_i - \bar{x}}{s} & x_{minmax} &= \frac{x - x_{min}}{x_{max} - x_{min}} & z &= \frac{x - \mu}{\sigma} \\ r_{xy} &= \frac{Cov(X, Y)}{\sigma_X \sigma_Y} & VIF_j &= \frac{1}{1 - R_j^2} & \Sigma &= \frac{1}{n-1} X^T X \\ EVR_k &= \frac{\lambda_k}{\sum_j \lambda_j} & r_t &= \frac{P_t - P_{t-1}}{P_{t-1}} & g_t &= \ln P_t - \ln P_{t-1} \end{aligned}$$

6.7 Final challenge: the 90-minute EDA interview simulation

You are handed an unknown customer dataset and told: “Find something useful.” You have 90 minutes. Your submission must include:

1. a one-paragraph problem framing with an explicit question;
2. a structural audit;
3. at least one data-quality finding that changes an analytical decision;
4. three targeted visualisations and written interpretations;
5. one transformation or feature with a documented reason;
6. one example of a tempting but invalid conclusion;
7. a concise recommendation and a list of what additional data would reduce uncertainty.

Think Like an Engineer

The strongest answer is not the notebook with the most plots. It is the notebook where every plot exists because a question existed before it.

7

Deep Practice Workbook — Unit 1

7.1 Worked example: from an unknown CSV to a data map

Imagine receiving a file called `sales.csv` with no documentation. Do not guess. First inspect the file mechanically, then ask semantic questions.

```
1 from pathlib import Path
2 import pandas as pd
3
4 path = Path("sales.csv")
5 print("Exists:", path.exists())
6 print("Size MB:", round(path.stat().st_size / 1024**2, 2))
7
8 df = pd.read_csv(path)
9 print("Shape:", df.shape)
10 print("Columns:", df.columns.tolist())
11 print("Dtypes:\n", df.dtypes)
12 print("Nulls:\n", df.isna().sum())
13 print("Duplicates:", df.duplicated().sum())
```

Think Like an Engineer

Notice the order. We did not calculate a correlation matrix. We first established whether the file exists, whether it loaded, how large it is, what the columns are, and whether obvious quality problems exist. Good EDA reduces uncertainty in the correct order.

7.2 Schema notebook pattern

Create a schema table that becomes the contract for the rest of the analysis.

```
1 schema = pd.DataFrame({
2     "column": df.columns,
3     "dtype": df.dtypes.astype(str).values,
4     "missing_pct": df.isna().mean().mul(100).round(2).values,
5     "unique_n": df.nunique(dropna=False).values,
6 })
7 schema["semantic_type"] = "review_required"
8 print(schema)
```

Then manually classify each column as identifier, target, numeric measurement, categorical label, datetime, free text, or derived field. The manual semantic pass is important because automated dtype inference cannot know business meaning.

7.3 Row-grain drill

Suppose the dataset contains `customer_id`, `order_id`, `product_id`, `quantity`. Ask: can one customer appear multiple times? Can one order contain multiple products? If yes, `order_id` is not a unique row key. This changes what “duplicate” means.

```
1 print("Unique customers:", df["customer_id"].nunique())
2 print("Unique orders:", df["order_id"].nunique())
3 print("Rows per order summary:")
4 print(df.groupby("order_id").size().describe())
```

7.4 Identifiers versus features

Identifiers can accidentally behave like numeric variables. An integer customer ID may have a large variance, but its magnitude has no physical meaning. Never feed identifiers into statistics or models merely because pandas says they are numeric.

Common Pitfalls & Edge Cases

A common beginner error is `df.corr(numeric_only=True)` followed by a statement such as “customer_id has low correlation, so it is unimportant.” The deeper conclusion is that correlation was the wrong operation for an identifier in the first place.

7.5 Distribution drill: same mean, different reality

Construct three arrays with approximately the same mean but very different shapes: symmetric, skewed and bimodal.

```
1 rng = np.random.default_rng(42)
2 a = rng.normal(50, 5, 1000)
3 b = rng.lognormal(mean=3.8, sigma=0.25, size=1000)
4 b = 50 * b / b.mean()
5 c = np.concatenate([
6     rng.normal(40, 3, 500),
7     rng.normal(60, 3, 500),
8 ])
9
10 for name, values in {"symmetric": a, "skewed": b, "bimodal": c}.items():
11     print(name, "mean=", values.mean(), "median=", np.median(values))
12
13 \section{Distribution drill: quantiles as a language}
14 Quantiles answer questions such as “What value is below 90% of observations”? They are often more useful
15     than a single standard deviation for skewed operational data.
16 \begin{lstlisting}
17 quantiles = df["amount"].quantile([0.01, 0.05, 0.25, 0.50, 0.75, 0.95, 0.99])
18 print(quantiles)
```

Write the result in words. For example: “The 99th percentile is ₹X, meaning approximately 99% of observed transactions are at or below X in this sample.”

7.6 Distribution drill: ECDF

The empirical cumulative distribution function avoids arbitrary histogram bins. For each threshold x , $\text{ECDF}(x)$ is the proportion of observations less than or equal to x .

```
1 def ecdf(values: np.ndarray) -> tuple[np.ndarray, np.ndarray]:
2     x = np.sort(values)
3     y = np.arange(1, len(x) + 1) / len(x)
4     return x, y
5
6 x, y = ecdf(df["amount"].dropna().to_numpy())
7 plt.plot(x, y)
```

```
8 plt.xlabel("Amount")
9 plt.ylabel("ECDF")
10 plt.title("Empirical distribution")
11 plt.show()
```

7.7 Distribution drill: Q-Q thinking

A Q-Q plot compares sample quantiles with theoretical quantiles. A roughly straight central region can support a normal approximation, while curved tails reveal departures. Do not use a Q-Q plot as a binary “normal/not normal” machine; interpret where and how it departs.

```
1 from scipy import stats
2
3 stats.probplot(df["amount"].dropna(), dist="norm", plot=plt)
4 plt.title("Normal Q-Q plot")
5 plt.show()
```

7.8 Unit 1 mini-project: dataset autopsy

Choose one public dataset. Produce exactly these outputs: a data card, schema table, row-grain statement, five data-quality findings, five semantic questions, three hypotheses and a one-page “what I still do not know” section. This teaches intellectual honesty before modelling.

8

Deep Practice Workbook — Unit 2

8.1 Central tendency worked calculation

For $x = [4, 5, 5, 6, 100]$,

$$\bar{x} = \frac{4 + 5 + 5 + 6 + 100}{5} = 24, \quad \text{median} = 5.$$

The mean is mathematically correct. The question is whether 24 is a useful description of a typical observation. This distinction between *correct calculation* and *useful interpretation* is central to EDA.

8.2 Variance worked calculation

For $x = [2, 4, 6]$, the mean is 4. Deviations are -2, 0, 2. Squared deviations are 4, 0, 4. Sample variance is $8/2 = 4$ and sample standard deviation is 2. Write the intermediate steps when learning; later, let NumPy perform the arithmetic but retain the conceptual chain.

```
1 x = np.array([2, 4, 6], dtype=float)
2 print("mean", x.mean())
3 print("sample variance", x.var(ddof=1))
4 print("sample std", x.std(ddof=1))
```

8.3 Why *ddof* matters

NumPy defaults and statistical conventions can differ. For sample standard deviation, use *ddof* = 1 when the conventional sample estimator is intended. Always know whether your library is computing population or sample variance.

8.4 Skewness formula and interpretation

One common moment-based population-style coefficient is

$$\gamma_1 = \frac{E[(X - \mu)^3]}{\sigma^3}.$$

The sign indicates asymmetry direction, but finite-sample estimators vary. In practice, combine the coefficient with a plot and quantiles rather than interpreting a threshold mechanically.

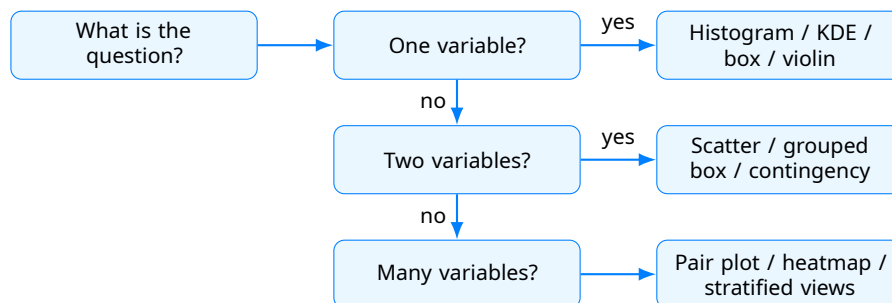
8.5 Kurtosis and tails

A fourth standardized central moment is

$$\beta_2 = \frac{E[(X - \mu)^4]}{\sigma^4}.$$

Excess kurtosis subtracts 3 under the normal-reference convention. Different libraries may report biased/unbiased forms. The practical question is often: “Are extreme observations more common than a light-tailed reference would suggest?”

8.6 Plot choice decision tree



8.7 Histogram bin sensitivity lab

Plot the same variable with 10, 20, 40 and 80 bins. Write which features remain stable across bin widths. Those stable features are usually more trustworthy than a single visual detail created by one arbitrary bin choice.

```

1 fig, axes = plt.subplots(2, 2, figsize=(11, 8))
2 for ax, bins in zip(axes.flat, [10, 20, 40, 80]):
3     sns.histplot(df["amount"].dropna(), bins=bins, ax=ax)
4     ax.set_title(f"Bins = {bins}")
5 plt.tight_layout()
6 plt.show()
  
```

8.8 Box plot versus violin plot

A box plot is excellent for compact comparison. A violin plot adds density shape. If two distributions have equal medians but one is bimodal, the violin may expose the hidden structure.

8.9 Outlier lab: four cases

Create four datasets: normal data with one typo, legitimate heavy-tailed data, two mixed populations, and bounded measurements. Apply Z-score and IQR to all four. Your task is to identify where each method produces misleading flags and explain why.

```

1 rng = np.random.default_rng(7)
2 cases = {
3     "typo": np.r_[rng.normal(50, 5, 199), 500],
4     "heavy_tail": rng.standard_t(df=3, size=200),
5     "mixture": np.r_[rng.normal(40, 3, 100), rng.normal(70, 3, 100)],
6     "bounded": rng.beta(2, 5, 200),
7 }
  
```

8.10 Missingness matrix interpretation

A missingness heatmap is a map of absence, not an explanation of cause. Add subgroup analysis and process knowledge. For example, a laboratory test may be missing because a clinician ordered it only for high-risk patients; that is informative selection, not random failure.

9

Deep Practice Workbook — Unit 3

9.1 Cleaning decision table

For every anomaly, classify it before changing it:

Observed issue	Possible meaning	Safe first action	Verification
Blank value	missing measurement / not applicable	profile mechanism	compare groups
-999	placeholder / true value	inspect documentation	count and range check
Duplicate key	repeated event / duplicate load	inspect business grain	compare timestamps
Mixed case	formatting inconsistency	normalise cautiously	unique-value audit
Impossible date	parser/data-entry issue	coerce + inspect failures	failure count + source check
Extreme number	rare valid / typo	investigate, do not auto-delete	source/domain validation

9.2 Imputation distortion demonstration

Generate a skewed variable, remove 30% of values, then compare mean imputation and median imputation. Measure the change in variance and quantiles. The point is not to choose one method universally; it is to observe the distortion each method creates.

```
1 rng = np.random.default_rng(42)
2 x = rng.lognormal(3.0, 0.8, 2000)
3 missing_mask = rng.random(len(x)) < 0.30
4 x_missing = x.copy()
5 x_missing[missing_mask] = np.nan
6
7 mean_filled = np.where(np.isnan(x_missing), np.nanmean(x_missing), x_missing)
8 median_filled = np.where(np.isnan(x_missing), np.nanmedian(x_missing), x_missing)
9
10 for name, values in [("original", x), ("mean", mean_filled), ("median", median_filled)]:
11     print(name, np.quantile(values, [0.25, 0.5, 0.75]), np.std(values))
```

9.3 Group-wise imputation reasoning

If salary distributions differ dramatically by job role, a single global median can be unrealistic. Group-wise imputation can preserve structure, but it also requires enough observations per group and careful handling of tiny groups.

```

1 df["salary"] = (
2     df.groupby("role")["salary"]
3     .transform(lambda s: s.fillna(s.median()))
4 )

```

Common Pitfalls & Edge Cases

The code above is not automatically safe in every modelling workflow. If it is executed before the train/test split, group medians can incorporate information from the held-out data. In model training, use a pipeline or fit the imputation logic on training data only.

9.4 Target encoding: leakage-safe mental model

Suppose a category has target mean 0.82. If you compute that mean using the entire dataset, the target values of validation rows influence the encoded value. The validation set is no longer unseen. Safe target encoding uses training folds to calculate category statistics and applies them to held-out rows.

9.5 Encoding selection table

Encoding	Best fit	Main risk	Mental test
One-hot	nominal, moderate cardinality	dimensionality	"Is category identity useful?"
Ordinal	truly ordered categories	fake distance	"Does order have meaning?"
Frequency	high-cardinality categories	frequency collisions	"Does prevalence carry signal?"
Target	predictive categorical signal	leakage	"Can I fit this only on training folds?"

9.6 Feature engineering: source-to-feature ledger

For each engineered feature, record:

1. source columns;
2. exact formula;
3. unit;
4. timestamp availability;
5. missing-data behaviour;
6. reason it might help;
7. leakage risk;
8. validation check.

9.7 Datetime engineering lab

Create features from a synthetic order table: weekday, month, hour, weekend flag, days since customer first order, and cyclical hour representation. Compare a raw hour feature with sine/cosine encoding conceptually and visually.

```
1 dt = pd.to_datetime(df["order_time"])
2 df = df.assign(
3     hour=dt.dt.hour,
4     weekday=dt.dt.dayofweek,
5     is_weekend=dt.dt.dayofweek.ge(5),
6     hour_sin=np.sin(2 * np.pi * dt.dt.hour / 24),
7     hour_cos=np.cos(2 * np.pi * dt.dt.hour / 24),
8 )
```

10

Deep Practice Workbook — Unit 4

10.1 Three correlation worlds

Build three datasets: linear, monotonic nonlinear and non-monotonic circular. Compute Pearson, Spearman and Kendall correlations. Explain why each coefficient succeeds or fails.

```
1 rng = np.random.default_rng(10)
2 x = np.linspace(-3, 3, 300)
3 y_linear = 2 * x + rng.normal(0, 0.3, len(x))
4 y_monotonic = np.exp(x) + rng.normal(0, 1.0, len(x))
5 t = np.linspace(0, 2*np.pi, 300)
6 x_circle = np.cos(t)
7 y_circle = np.sin(t)
```

10.2 Correlation is not causation lab

Construct a synthetic “ice cream sales” and “sunburn” dataset where temperature drives both. Compute correlation, draw a scatter plot, then stratify by temperature bands. Write why the original relationship should not be interpreted as “ice cream causes sunburn.”

10.3 Simpson’s paradox exercise

Create two groups where X and Y are positively associated within each group but negatively associated after aggregation. This forces you to understand why subgroup structure matters. The exercise should be solved by constructing the data rather than searching for an example.

10.4 VIF worked example

Suppose $x_3 = 2x_1 + \epsilon$. Regressing x_3 on x_1 and other predictors yields a high R^2 , so $1/(1 - R^2)$ becomes large. The important idea is not the threshold; it is that one coefficient becomes difficult to interpret because the model has trouble separating overlapping predictors.

10.5 PCA from scratch on a two-dimensional dataset

Use NumPy to compute the covariance matrix, eigenvalues and eigenvectors manually, then compare with scikit-learn.

```

1 X = np.array([
2     [2.0, 1.0],
3     [3.0, 2.0],
4     [4.0, 3.1],
5     [5.0, 3.9],
6     [6.0, 5.2],
7 ])
8 Xc = X - X.mean(axis=0)
9 cov = np.cov(Xc, rowvar=False)
10 eigenvalues, eigenvectors = np.linalg.eigh(cov)
11 order = np.argsort(eigenvalues)[::-1]
12 eigenvalues = eigenvalues[order]
13 eigenvectors = eigenvectors[:, order]
14 print("Covariance:\n", cov)
15 print("Eigenvalues:", eigenvalues)
16 print("Eigenvectors:\n", eigenvectors)

```

Think Like an Engineer

The largest eigenvalue belongs to the direction that captures the greatest variance. The sign of an eigenvector is arbitrary: v and $-v$ describe the same axis. Students often think a negative loading means “negative importance”; that is not generally the correct interpretation.

10.6 PCA reconstruction thought experiment

Project the data onto one component, then reconstruct it. The reconstruction error is the information discarded by the dimensionality reduction under the chosen squared-error geometry.

```

1 from sklearn.decomposition import PCA
2 from sklearn.preprocessing import StandardScaler
3
4 X_scaled = StandardScaler().fit_transform(X)
5 pca = PCA(n_components=1)
6 Z = pca.fit_transform(X_scaled)
7 X_reconstructed = pca.inverse_transform(Z)
8 reconstruction_error = np.mean((X_scaled - X_reconstructed) ** 2)
9 print("Reconstruction MSE:", reconstruction_error)

```

10.7 Automated EDA: what the machine can and cannot decide

Automated profiling is excellent for breadth: it can flag missingness, duplicates, distributions and relationships quickly. But alerts need domain validation. A high-cardinality identifier may be flagged as categorical; a rare but valid clinical value may be called an outlier; a strong correlation may be an artefact of a common time trend. Automation accelerates inspection; it does not replace judgement.

10.8 Dashboard design lab

Build a Streamlit dashboard with five zones: dataset health, feature explorer, relationship explorer, time explorer, and findings/limitations. Add filters only when they answer a real analytical question. Avoid a dashboard containing twenty selectors and no narrative.

11

Deep Practice Workbook — Unit 5 and Capstone

11.1 Healthcare scenario

You receive a synthetic hospital dataset with age, sex, admission type, length of stay, lab values and outcome. Your first task is not classification. Profile imbalance, missingness, impossible values, repeated patients and subgroup outcome rates. Then ask whether missing laboratory values are related to severity or care pathway.

```
1 summary = (  
2     df.groupby("admission_type", dropna=False)  
3     .agg(  
4         patients=("patient_id", "nunique"),  
5         rows=("patient_id", "size"),  
6         outcome_rate=("outcome", "mean"),  
7     )  
8     .sort_values("outcome_rate", ascending=False)  
9 )  
10 print(summary)
```

11.2 Retail scenario

You receive order-level data. Before revenue analysis, determine whether refunds, cancellations and returned quantities are represented. A naive sum of line-item revenue can overstate realised revenue if the business definition is net sales.

```
1 df["gross_value"] = df["quantity"] * df["unit_price"]  
2 df["net_value"] = df["gross_value"] - df["refund_amount"].fillna(0)  
3  
4 monthly = (  
5     df.assign(month=df["order_date"].dt.to_period("M"))  
6     .groupby("month")["net_value"]  
7     .sum()  
8 )
```

11.3 Finance scenario

For market data, check missing trading days, corporate actions if applicable, duplicated timestamps, price/volume anomalies and whether the analysis uses adjusted or unadjusted values. Then calculate returns and rolling volatility. Never infer a stable average from a non-stationary price series without checking the time structure.

11.4 Capstone sprint plan

Sprint	Deliverable	Checkpoint
1	Problem framing + data card	Can another student state the question?
2	Structural audit + schema	Are row grain and keys understood?
3	Distribution report	Are skew, tails and missingness explained?
4	Relationship report	Are subgroup differences considered?
5	Cleaning contract	Can every transformation be defended?
6	Feature ledger	Is every feature available at decision time?
7	Dashboard/report	Can a stakeholder find the key insight?
8	Final challenge review	Can you reproduce the results from a clean run?

11.5 Capstone failure injection

A strong student should learn to detect failures deliberately. Inject these problems one at a time: duplicate rows, impossible ages, a -999 placeholder, mixed date formats, a target leak, extreme outliers, category spelling variants, 30% missingness, and a train/test preprocessing leak. Ask another student to find them without seeing the answer key.

11.6 Final practical checklist

Before declaring an EDA complete, answer all of the following:

1. What exactly is one row?
2. Where did the data come from?
3. What period does it cover?
4. Which columns are identifiers?
5. Which values are impossible?
6. How much data is missing, and is the missingness informative?
7. Are there duplicates at the business grain?
8. Which variables are skewed or heavy-tailed?
9. Which relationships are linear, monotonic, nonlinear or subgroup-specific?
10. Which features could leak future information?
11. Which transformations are justified by the question?
12. Did every fitted preprocessing step use training data only?
13. What changed after cleaning?
14. What did the analysis fail to establish?
15. What should the team do next?

11.7 Answer-writing practice

Take this weak statement: “Income is correlated with churn.” Rewrite it into an evidence-based statement containing direction, strength, sample scope, subgroup qualification and a limitation. The goal is to replace vague certainty with precise communication.

11.8 Student self-improvement loop

After every lab, maintain a four-column journal:

I got wrong	Why it happened	How I detected it	Rule for next time
...	...	...	...

The journal is deliberately about mistakes. Technical growth comes from converting repeated mistakes into reusable mental rules.

12

Master Question Bank

12.1 Conceptual questions

1. Define EDA and distinguish it from modelling.
2. Explain nominal, ordinal, interval and ratio scales with examples.
3. Why is measurement scale more important than storage dtype?
4. What is row grain and why does it matter?
5. Explain mean, median, mode and trimmed mean.
6. Derive sample variance.
7. Explain IQR and Tukey fences.
8. Explain MCAR, MAR and MNAR.
9. Compare mean, median and group-wise imputation.
10. Why can target encoding leak?
11. Compare one-hot and ordinal encoding.
12. Explain interaction features.
13. Explain Pearson, Spearman and Kendall.
14. Derive the VIF formula conceptually.
15. Explain covariance as a matrix.
16. Derive the PCA eigenvalue problem.
17. Explain explained variance ratio.
18. Why should PCA often follow scaling?
19. Why is automated profiling not a final analysis?
20. How does domain context change EDA?

12.2 Coding questions

1. Write a function that reports missing percentage by column.
2. Write a function that reports duplicate counts for a supplied business key.

3. Parse a datetime column and create weekday/hour features.
4. Plot a numeric distribution with histogram and KDE.
5. Compare two groups using box and violin plots.
6. Flag IQR outliers without deleting them.
7. Build a ColumnTransformer for mixed numeric/categorical data.
8. Create a leakage-safe train/test preprocessing pipeline.
9. Compute Pearson, Spearman and Kendall for two columns.
10. Produce a PCA scree plot and cumulative variance plot.
11. Build a minimal Streamlit feature explorer.

12.3 Debugging questions

```
1 # 1. Why might this fail or produce unexpected alignment?
2 df["new"] = pd.Series([1, 2, 3])
3
4 # 2. What is wrong with fitting this before the split?
5 scaler.fit_transform(X)
6
7 # 3. Why might this statistic be misleading?
8 df.groupby("customer_id")["revenue"].mean().mean()
9
10 # 4. Why is this a dangerous feature for Friday prediction?
11 df["refund_after_cancel"]
```

12.4 Design questions

1. Design an EDA plan for a hospital readmission dataset.
2. Design an EDA plan for an e-commerce cohort dataset.
3. Design an EDA plan for a stock-return dataset.
4. Design a dashboard for a founder who wants five actionable metrics and two anomaly views.
5. Design a data-quality contract for a production CSV feed.

13

First-Semester Python Bridge for EDA

13.1 What Python knowledge is actually required?

EDA is easier when a student knows a small, reliable subset of Python rather than trying to learn the whole language first. You need variables, basic types, lists/dictionaries, conditions, loops for understanding, functions, imports, exceptions, and reading documentation. Most EDA work then becomes structured manipulation of arrays and tables.

Core Definition

Variable: a name referring to a value. **Function:** reusable behaviour that accepts inputs and may return an output. **Library:** reusable code written by others. **DataFrame:** a labelled two-dimensional table provided by Pandas.

13.1.1 The smallest useful Python example

```
1 marks = [72, 81, 65, 90]
2 average = sum(marks) / len(marks)
3 print(average)
```

Think Like an Engineer

Ask yourself: What does `sum(marks)` do? What does `len(marks)` do? Why is division valid here? If you can answer these three questions, you are already building the reasoning needed for Pandas aggregation.

13.1.2 Conditions: turning observations into decisions

```
1 attendance = 78
2
3 if attendance >= 75:
4     status = "eligible"
5 else:
6     status = "check policy"
7
8 print(status)
```

The same logic later appears in EDA as a data-quality rule: a percentage below 0 or above 100 should trigger investigation.

13.1.3 Functions: package your thinking

```

1 from typing import Iterable
2
3
4 def mean_of(values: Iterable[float]) -> float:
5     values = list(values)
6     if not values:
7         raise ValueError("values cannot be empty")
8     return sum(values) / len(values)
9
10 print(mean_of([10, 20, 30]))

```

Common Pitfalls & Edge Cases

Do not confuse function with function call. `df.describe` refers to a method object; `df.describe()` calls it. This tiny difference is one of the most common beginner bugs.

13.2 Reading an error instead of fearing it

When Python reports an error, read the final line first, then move upward through the traceback to the line in your own code.

```

1 marks = [70, 80, 90]
2 print(marks.mean())

```

This fails because a normal Python list does not have the Pandas Series method `mean`. The useful response is not “Python is difficult”; it is “What object do I have, and what operations does that object support?”

Hands-On Lab Pattern

Convert the list to a NumPy array and then a Pandas Series. Compare `type(...)` and available operations. Repeat with a DataFrame column. Record the differences in your own notebook.

13.3 EDA coding conventions from day one

```

1 # Good: descriptive names and one clear operation per line.
2 students = pd.read_csv("students.csv")
3 students["marks"] = pd.to_numeric(students["marks"], errors="coerce")
4
5 # Avoid: unexplained one-letter variables in a serious analysis.
6 x = pd.read_csv("students.csv")

```

Industry Practice

A junior engineer is often judged less by how clever their code looks and more by whether another person can understand, reproduce and safely modify it. Use descriptive names, small functions, assertions, comments for decisions, and deterministic steps.

14

EDA Laboratory Patterns You Should Reuse

14.1 The universal first-pass notebook

The following pattern is deliberately simple. It should become a mental template, not a blind copy.

```
1 from pathlib import Path
2 import numpy as np
3 import pandas as pd
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6
7 sns.set_theme(style="whitegrid", context="notebook")
8
9 DATA_PATH = Path("data.csv")
10 df = pd.read_csv(DATA_PATH)
11
12 print("Shape:", df.shape)
13 print("\nColumns:")
14 print(df.columns.tolist())
15 print("\nDtypes:")
16 print(df.dtypes)
17 print("\nSample:")
18 print(df.head())
19 print("\nMissing values:")
20 print(df.isna().sum().sort_values(ascending=False))
21 print("\nDuplicate rows:", df.duplicated().sum())
22 print("\nNumeric summary:")
23 print(df.describe(numeric_only=True).T)
```

Think Like an Engineer

Notice that this script does not start with a fancy chart. It first asks: How large is the dataset? What are the columns? What types are they? Are values missing? Are rows duplicated? Only then do we move to deeper analysis.

14.2 A reusable data-quality report function

```
1 from typing import Any
2 import pandas as pd
3
4
5 def quality_report(df: pd.DataFrame) -> pd.DataFrame:
6     """Return one row per column with basic quality diagnostics."""
7     report = pd.DataFrame({
8         "dtype": df.dtypes.astype(str),
9         "missing": df.isna().sum(),
```

```

10     "missing_pct": df.isna().mean().mul(100),
11     "unique": df.nunique(dropna=True),
12 })
13 report["sample_value"] = [
14     df[column].dropna().iloc[0] if df[column].notna().any() else None
15     for column in df.columns
16 ]
17 return report.sort_values("missing_pct", ascending=False)
18
19
20 report = quality_report(df)
21 print(report)

```

14.2.1 Why this is better than a giant one-liner

A function gives a repeatable contract. You can test it, document it, reuse it across datasets, and improve it without rewriting every notebook.

14.3 Assertions: turn assumptions into executable checks

```

1 assert not df.empty, "Dataset is empty"
2 assert df.columns.is_unique, "Column names must be unique"
3
4 if "age" in df.columns:
5     assert df["age"].dropna().between(0, 120).all(), \
6         "Age contains impossible values"

```

Industry Practice

Assertions are especially valuable when an analysis becomes a recurring job. If the upstream dataset changes unexpectedly, failing early is safer than silently producing a plausible-looking report.

14.4 Groupby: one of the most important EDA ideas

Suppose the dataset contains department and marks. We may ask whether the distribution differs by department.

```

1 summary = (
2     df.groupby("department", observed=True)["marks"]
3     .agg(["count", "mean", "median", "std"])
4     .sort_values("mean", ascending=False)
5 )
6 print(summary)

```

The code is less important than the question: *What grouping makes the comparison meaningful?* Grouping by a meaningless identifier can produce a table that is technically correct but analytically useless.

14.5 Filtering: translate English into boolean logic

```

1 # Students with marks below 40 and attendance below 75.
2 flagged = df.loc[
3     (df["marks"] < 40) & (df["attendance"] < 75)
4 ]

```

Common Pitfalls & Edge Cases

Use & and | with Pandas boolean Series, not Python's and or. Parenthesise each comparison. This is both a syntax and logic issue.

14.6 Vectorisation: think in columns

```
1 # Vectorised feature.  
2 df["pass"] = df["marks"] >= 40  
3  
4 # Vectorised arithmetic.  
5 df["weighted_score"] = 0.7 * df["exam"] + 0.3 * df["assignment"]
```

For ordinary column operations, vectorisation is clearer and usually faster than writing a Python loop over every row.

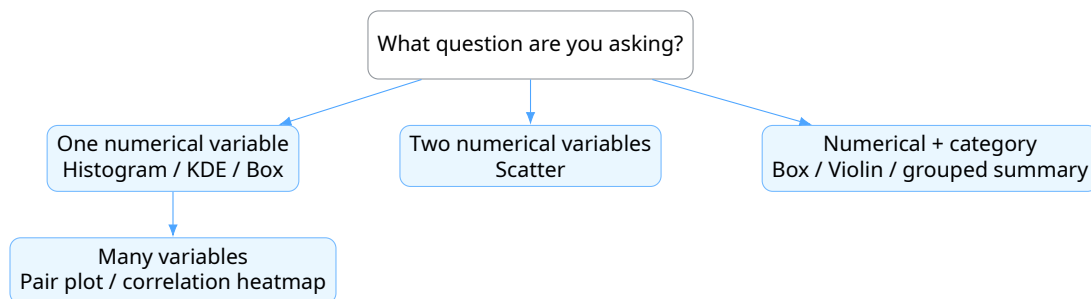
14.7 When a loop is actually useful

Loops are useful for control flow, file processing, iterative algorithms and situations where vectorisation is awkward. The rule is not “never use loops”; the rule is “do not manually loop through table rows when a clear vectorised operation exists.”

15

Visual Reasoning Workbook

15.1 Plot selection decision tree



15.2 Histogram: a measurement of your plotting choices

A histogram is not “the distribution”. It is an approximation of a distribution using bins. Different bin widths can reveal or hide structure.

```
1 fig, axes = plt.subplots(1, 3, figsize=(14, 4))
2 for ax, bins in zip(axes, [5, 15, 40]):
3     ax.hist(df["marks"].dropna(), bins=bins, edgecolor="black")
4     ax.set_title(f"bins={bins}")
5     ax.set_xlabel("Marks")
6     ax.set_ylabel("Count")
7 plt.tight_layout()
8 plt.show()
```

Think Like an Engineer

If a conclusion exists only for one arbitrary bin setting, be cautious. Change the bin width and ask whether the story survives.

15.3 Box plot: understand the five-number story

A conventional box plot communicates median, quartiles, spread and potential outliers. It does not prove that every point outside the whisker is an error.

15.4 Scatter plot: relationship, not proof of cause

A visible upward pattern suggests association. It does not establish causality. Add colour, faceting or a third variable only when doing so answers a meaningful question.

```
1 sns.scatterplot(data=df, x="attendance", y="marks", hue="department")
2 plt.title("Marks versus attendance by department")
3 plt.xlabel("Attendance (%)")
4 plt.ylabel("Marks")
5 plt.show()
```

15.5 ECDF: a powerful beginner-friendly distribution tool

The empirical cumulative distribution function answers: “What fraction of observations are less than or equal to this value?”

```
1 values = np.sort(df["marks"].dropna().to_numpy())
2 y = np.arange(1, len(values) + 1) / len(values)
3
4 plt.step(values, y, where="post")
5 plt.xlabel("Marks")
6 plt.ylabel("Fraction <= x")
7 plt.title("Empirical CDF of marks")
8 plt.show()
```

16

Statistics Without Memorisation

16.1 Why different summaries answer different questions

Use a deliberately uneven dataset:

10, 10, 11, 11, 12, 12, 13, 100.

The mean is pulled upward by the extreme value; the median is much more stable. This is not a rule that “median is always better”. It is a reminder that a statistic encodes a particular sensitivity to the data.

16.2 Sample variance: derive it from deviation

Start with the mean $\bar{x}$. Each observation has deviation $x_i - \bar{x}$. Deviations can cancel, so square them:

$$\sum_{i=1}^n (x_i - \bar{x})^2.$$

For a sample, divide by $n - 1$:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2.$$

The square root gives sample standard deviation.

Think Like an Engineer

The $n - 1$ denominator is not a magic programming convention. It is the degrees-of-freedom correction used when estimating population variance from a sample after estimating the mean from that same sample.

16.3 IQR and robust spread

$$IQR = Q_3 - Q_1.$$

The common exploratory fences are:

$$L = Q_1 - 1.5IQR, \quad U = Q_3 + 1.5IQR.$$

These are *flagging rules*, not universal laws about invalid observations.

16.4 Skewness: reason about the tail

A right-skewed distribution often has a long upper tail and can satisfy approximately:

$$\text{mean} > \text{median}.$$

Do not turn that approximation into a theorem for every finite sample.

16.5 Kurtosis: avoid the “peakedness only” trap

Kurtosis is strongly related to tail behaviour. Different software conventions may report raw or excess kurtosis. Always check the definition before comparing numbers across tools.

16.6 Trimmed mean: a deliberate compromise

A trimmed mean removes a fixed fraction of the lowest and highest observations before averaging. It can reduce sensitivity to extremes while retaining more numerical information than the median.

17

Missingness, Bias and Data Quality Reasoning

17.1 Missing is a state, not a number

A missing value should not automatically become zero. Zero means something specific in many domains. Missing means “the value is unavailable under the current data process.”

17.2 A missingness investigation

```
1 missing = (  
2     df.isna()  
3     .mean()  
4     .mul(100)  
5     .sort_values(ascending=False)  
6     .rename("missing_pct")  
7 )  
8 print(missing)  
9  
10 # Compare missingness by a relevant grouping variable.  
11 if "department" in df.columns:  
12     print(df.groupby("department", observed=True).apply(  
13         lambda g: g.isna().mean(), include_groups=False  
14     ))
```

The second analysis is important because missingness may not be evenly distributed across groups.

17.3 MCAR, MAR and MNAR: a practical mental model

- | | |
|-------------|---|
| MCAR | Missingness is unrelated to observed and unobserved values. Rare and strong assumptions. |
| MAR | Missingness can be explained using variables that are observed. Group comparisons can provide evidence. |
| MNAR | Missingness depends on the missing value itself or an unobserved process. Hard to verify from the observed table alone. |

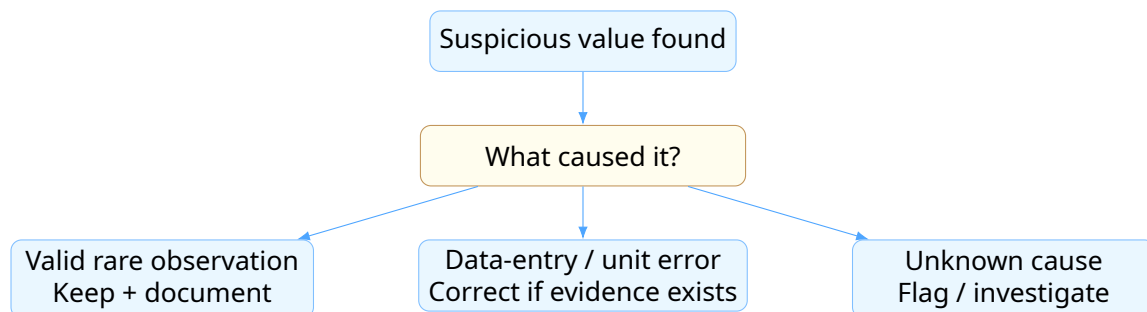
Common Pitfalls & Edge Cases

Students often try to “detect MAR” or “prove MNAR” from one missingness percentage. That is not enough. These are assumptions about the data-generating mechanism, not labels assigned by a single function call.

18

Cleaning Without Destroying Evidence

18.1 The cleaning decision tree



18.2 Duplicates: exact duplicate versus entity duplicate

```
1 # Exact row duplicates.
2 exact_duplicates = df.duplicated(keep=False)
3
4 # Possible duplicate student records based on a business key.
5 possible_duplicates = df.duplicated(
6     subset=["student_id", "exam_date"], keep=False
7 )
```

A duplicate is defined relative to the meaning of a row. Two identical rows may be accidental duplicates, but two rows with the same student and date might legitimately represent two events. Never delete rows solely because a duplicate check returned True.

18.3 Structural inconsistencies

Typical examples include:

- Indore, indore, Indore and INDORE representing one category.
- Dates mixed as 2026-01-05, 05/01/2026 and spreadsheet serial values.
- Numeric strings containing commas, currency symbols or accidental text.
- Units mixed between centimetres and metres, rupees and thousands of rupees, Celsius and Fahrenheit.

```
1 city = (
2     df["city"].astype("string")
3     .str.strip()
4     .str.casefold())
```

```
5 )  
6 city_map = {"indore": "Indore", "bhopal": "Bhopal"}  
7 df["city"] = city.map(city_map).fillna(city.str.title())
```

19

Feature Engineering: From Raw Columns to Useful Signals

19.1 Feature engineering starts with a question

A feature should exist because it represents useful information, not because creating more columns feels like progress.

19.1.1 Interaction features

If performance may depend jointly on attendance and study hours, an interaction can represent the combined effect:

$$z = \text{attendance} \times \text{study_hours}.$$

```
1 df["attendance_study_interaction"] = (  
2     df["attendance"] * df["study_hours"]  
3 )
```

19.1.2 Polynomial features

For a numerical variable x , second-order expansion may include x and x^2 . For two variables, interactions such as x_1x_2 can appear. The important idea is that feature expansion can increase model flexibility while also increasing dimensionality and overfitting risk.

19.1.3 Datetime features

A timestamp is rarely useful as a raw string. Parse it first.

```
1 df["timestamp"] = pd.to_datetime(  
2     df["timestamp"], errors="coerce", utc=True  
3 )  
4 df["year"] = df["timestamp"].dt.year  
5 df["month"] = df["timestamp"].dt.month  
6 df["day_of_week"] = df["timestamp"].dt.dayofweek  
7 df["hour"] = df["timestamp"].dt.hour
```

19.1.4 Cyclical time

Hour 23 and hour 0 are close in time even though their numeric values are far apart. A cyclical encoding uses:

$$\sin(2\pi t/T), \quad \cos(2\pi t/T).$$

This is a useful extension when modelling periodic behaviour.

19.2 Feature-source ledger

For every engineered feature, record:

Feature		Source	Reason / availability rule
attendance	ex-	attendance + study hours	available before prediction time
tunderscore			
study		timestamp	derived from event time
month		historical values	must exclude future observations
rolling	extunder-		
score mean			

Common Pitfalls & Edge Cases

A mathematically correct feature can still be invalid if it uses information that would not have been available at prediction time. This is temporal leakage.

20

Encoding and Scaling: Choosing Rather Than Memorising

20.1 Encoding decision table

Data situation	Typical method	Main caution
Nominal, few categories	One-hot	Can expand dimensionality.
Ordered categories	Ordinal	Numeric ordering must have meaning.
High-cardinality category	Frequency / carefully designed target encoding	Leakage and unstable rare categories.
Continuous numerical feature	Standardisation / robust scaling	Fit parameters on training data only for predictive workflows.
Strongly skewed positive feature	Log / power transform	Check zeros, negatives and interpretability.

20.2 Min-Max scaling

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}.$$

It maps the observed range to approximately $[0, 1]$. It is sensitive to extreme minimum/maximum values.

20.3 Z-score standardisation

$$z = \frac{x - \mu}{\sigma}.$$

It centres around zero and scales by standard deviation. For predictive modelling, estimate μ and σ from training data only.

20.4 L2 row normalisation

For vector x :

$$\hat{x} = \frac{x}{\sqrt{x_1^2 + x_2^2 + \dots + x_p^2}}.$$

This changes the length of each row vector rather than independently standardising columns. That distinction is essential.

20.5 Robust scaling

A common robust transformation uses median and IQR:

$$x' = \frac{x - \text{median}(x)}{IQR(x)}.$$

It is often less influenced by extreme values than mean/std scaling.

21

Correlation, Covariance and Multivariate Thinking

21.1 Pearson correlation from standardised co-movement

For paired observations (x_i, y_i) :

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}.$$

The numerator measures whether deviations move together. The denominator scales the result.

21.2 Spearman and Kendall

Spearman correlation is Pearson correlation applied to ranks. It is useful for monotonic relationships that may not be linear. Kendall's tau is based on concordant and discordant pairs and has a different interpretation and sampling behaviour.

Think Like an Engineer

Before selecting a correlation coefficient ask: Are both variables numerical? Is the relationship roughly linear or merely monotonic? Are there severe outliers? Are ties common? The coefficient should follow the data structure, not the other way around.

21.3 Covariance matrix

For centred feature matrix X with n observations:

$$\Sigma = \frac{1}{n-1} X^T X.$$

Diagonal entries are variances; off-diagonal entries are covariances.

21.4 VIF: why multicollinearity matters

For predictor j :

$$VIF_j = \frac{1}{1 - R_j^2},$$

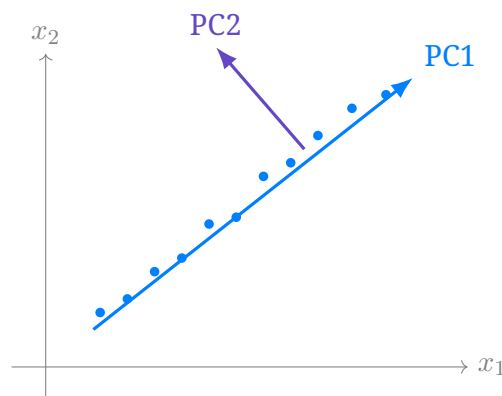
where R_j^2 comes from regressing predictor j on the remaining predictors. As R_j^2 approaches 1, the predictor is increasingly explainable by the others and its unique contribution becomes harder to separate in coefficient-based models.

22

PCA — From Geometry to Mathematics to Code

22.1 The two-dimensional picture

Imagine points forming an elongated cloud. The first principal component points along the direction with the greatest variance. The second component is orthogonal to it and captures the next largest remaining variance.



22.2 Eigenvectors and eigenvalues

For covariance matrix Σ , PCA solves:

$$\Sigma v = \lambda v.$$

The eigenvector v is a direction; the eigenvalue λ tells how much variance lies along that direction. Sort eigenvalues from largest to smallest to rank principal components.

22.3 Variance retention

If eigenvalues are $\lambda_1, \dots, \lambda_p$, explained variance ratio of component k is:

$$EV R_k = \frac{\lambda_k}{\sum_{j=1}^p \lambda_j}.$$

Cumulative explained variance helps decide how many components preserve a chosen fraction of variance.

22.4 PCA in code

```
1 from sklearn.preprocessing import StandardScaler
2 from sklearn.decomposition import PCA
3
4 features = df[["height", "weight", "study_hours", "marks"]]
5 X = StandardScaler().fit_transform(features)
6
7 pca = PCA(n_components=0.95, random_state=0)
8 X_reduced = pca.fit_transform(X)
9
10 print("Components retained:", pca.n_components_)
11 print("Explained variance ratio:", pca.explained_variance_ratio_)
12 print("Cumulative:", pca.explained_variance_ratio_.cumsum())
```

Common Pitfalls & Edge Cases

Do not compare PCA results across differently scaled features without thinking. A feature measured in large numerical units can dominate covariance-based PCA. Standardisation is often appropriate when features are on different scales, but it is still a modelling decision that should be justified.

23

Automated EDA and Dashboards — Use the Machine, Keep the Judgment

23.1 What automated EDA is good at

Automated tools can rapidly produce missingness summaries, type guesses, distributions, correlations, warnings and basic statistics. This is excellent for triage.

23.2 What automated EDA cannot decide reliably

It cannot know from a column name alone whether a value is a valid rare event, a unit error, a business-rule violation, a measurement artefact or a legitimate change in the process. Domain context remains necessary.

23.3 A minimal Streamlit EDA dashboard

```
1 import streamlit as st
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5
6 st.set_page_config(page_title="EDA Explorer", layout="wide")
7 st.title("EDA Explorer")
8
9 uploaded = st.file_uploader("Upload a CSV", type=["csv"])
10
11 if uploaded is not None:
12     df = pd.read_csv(uploaded)
13     st.write("Shape:", df.shape)
14     st.dataframe(df.head(20), use_container_width=True)
15
16     numeric = df.select_dtypes(include="number").columns.tolist()
17     if numeric:
18         column = st.selectbox("Choose a numeric column", numeric)
19         fig, ax = plt.subplots(figsize=(8, 4))
20         sns.histplot(df[column].dropna(), kde=True, ax=ax)
21         ax.set_title(f"Distribution of {column}")
22         st.pyplot(fig)
```

Industry Practice

A dashboard is a communication layer, not a replacement for an analysis notebook. Keep raw inputs, cleaning decisions, code and conclusions reproducible outside the UI.

24

Domain Thinking Workbook

24.1 Healthcare

Questions should include: Is the outcome imbalanced? Are measurements collected at different times? Are there protected or sensitive fields? Is the data minimised? Could an apparent pattern be caused by treatment-selection differences?

Hands-On Lab Pattern

Build a synthetic patient dataset with 1,000 rows. Include age, glucose, blood pressure, BMI, smoking status and a rare outcome. Perform: schema inspection, missingness profiling, class counts, distributions, outlier investigation, subgroup comparisons, correlation analysis and a short limitations report. Do not attempt to identify individuals or include real patient information.

24.2 Retail

Retail EDA often needs time-aware thinking. Daily sales may be affected by weekends, festivals, promotions, store opening dates, stockouts and changing product assortments.

```
1 df["date"] = pd.to_datetime(df["date"], errors="coerce")
2 df["weekday"] = df["date"].dt.day_name()
3 df["month"] = df["date"].dt.month
4 df["revenue"] = df["units"] * df["unit_price"]
```

24.3 Customer cohorts

A cohort groups customers by a meaningful starting event, such as first purchase month. Compare later behaviour by cohort rather than mixing all customers together.

24.4 Finance

Returns are often more informative than raw prices for many analyses:

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}.$$

Log returns are:

$$\ell_t = \ln \left(\frac{P_t}{P_{t-1}} \right).$$

Volatility is time-varying in many financial series, so a single global standard deviation can hide important regimes.

Common Pitfalls & Edge Cases

Financial time series violate many naive assumptions. Random train/test splitting can leak future information into the past. Always think about chronological order when the task depends on time.

25

Job-Ready EDA: From Notebook to Portfolio

25.1 What an internship interviewer may actually ask

1. You receive a CSV you have never seen. What are your first ten checks?
2. How do you decide whether a missing value should be imputed or left missing?
3. Why can mean and median tell different stories?
4. When would you choose a box plot over a histogram?
5. How can an outlier be useful information?
6. What is data leakage? Give a preprocessing example.
7. Why can correlation be misleading?
8. Explain PCA without using the word “dimension” first.
9. What would make your EDA non-reproducible?
10. How would you communicate a surprising finding to a non-technical manager?

25.2 A portfolio-quality EDA repository

A simple structure is:

```
1 eda-project/|—
2   README.md|—
3   data/|
4     |— raw/|
5     |— processed/|—
6   notebooks/|
7     |— 01_profile.ipynb|
8     |— 02_clean.ipynb|
9     |— 03_relationships.ipynb|—
10  src/|
11    |— io.py|
12    |— quality.py|
13    |— features.py|—
14  reports/|
15    |— eda_summary.pdf|—
16  tests/|
17    |— test_quality.py|—
18  requirements.txt
```

Industry Practice

A portfolio project becomes stronger when the README states the question, dataset source, data dictionary, cleaning decisions, key findings, limitations, how to reproduce the work, and what you would do next. A screenshot of ten plots is not a substitute for an argument.

25.3 EDA report template

Every final report should answer:

1. What problem are we investigating?
2. What does one row represent?
3. Where did the data come from?
4. What quality problems were found?
5. What distributions matter?
6. What relationships are supported by evidence?
7. Which observations are uncertain or suspicious?
8. What transformations were applied and why?
9. What conclusions are safe?
10. What cannot be concluded from this analysis?
11. What should happen next?

26

Exam Mastery and Logic-Building Workbook

26.1 How to write a strong theory answer

For a 5–10 mark question use this sequence:

1. Definition in one precise sentence.
2. Intuition or purpose.
3. Formula/algorithm if applicable.
4. Small example.
5. Diagram/table when helpful.
6. Advantages/limitations or edge cases.
7. Python implementation when asked.
8. One concluding sentence connecting back to EDA.

26.2 Example: “Explain outlier detection using IQR”

A high-quality answer should state that the IQR is $Q_3 - Q_1$, define the exploratory fences, show the formula, explain that a flagged observation is not automatically erroneous, and demonstrate the calculation on a small ordered dataset. A weak answer only writes “values outside 1.5 IQR are outliers.”

26.3 Code-to-logic translation drills

For each snippet, explain the English meaning before explaining the Python.

```
1 flag = df["age"].lt(0) | df["age"].gt(120)
```

English:

Select rows whose age is less than zero OR greater than 120.

```
1 summary = df.groupby("city")["salary"].median()
```

English:

For each city, calculate the median salary.

Think Like an Engineer

If you can translate code into precise English, and translate a precise English requirement back into code, your programming ability is becoming transferable rather than memorised.

26.4 Debugging ladder

When code fails, use this order:

1. Read the final error message.
2. Inspect the failing line.
3. Print or inspect the object's type.
4. Inspect shape and columns.
5. Check a tiny example.
6. Reproduce the issue in isolation.
7. Fix the smallest cause.
8. Re-run the original workflow.

26.5 Find the bug — mixed types

```
1 df["marks"] = ["70", "80", "absent", "90"]
2 print(df["marks"].mean())
```

Question: What is the data problem? How would you convert it safely? What information might be lost?

```
1 df["date"] = pd.to_datetime(df["date"])
2 df["month"] = df["date"].dt.month
```

Question: What happens if some date strings are malformed? When would you use `errors="coerce"`, and what should you inspect afterwards?

26.6 Find the bug — chained assumptions

```
1 train = df.sample(frac=0.8, random_state=42)
2 mean = df["income"].mean()
3 train["income"] = train["income"].fillna(mean)
```

The code may run, but for a predictive workflow the imputation statistic was estimated using the entire dataset, allowing information from the eventual validation/test portion to influence preprocessing. The better design fits the imputer using training data only and applies it to held-out data.

27

Independent Practice: Build Your Own Logic

27.1 The “no function name” challenge

Do these tasks without searching for a function name first. Write the algorithm in plain English, then implement it.

1. Find the percentage of missing values in each column.
2. Identify rows containing an impossible age.
3. Compare median marks across departments.
4. Find the top five columns by unique-value count.
5. Compare the distribution of a numerical variable before and after removing one extreme observation.
6. Detect exact duplicate rows and report their count.
7. Create a new feature representing total assessment score.
8. Produce a plot that answers whether two numerical variables move together.

27.2 The “predict the output” challenge

Before running:

```
1 s = pd.Series([10, 20, 20, 30])
2 print(s.mean())
3 print(s.median())
4 print(s.mode().tolist())
```

Write the expected output first. Then execute. If wrong, explain exactly why.

27.3 The “change one thing” challenge

Take any notebook in this book and change exactly one of:

- sample size,
- outlier magnitude,
- missingness rate,

- number of categories,
- skewness,
- group sizes.

Predict which statistic or plot should change most. Then verify it. This is a simple but powerful experiment-design habit.

28

Final Integrated Practical — The Student EDA Mission

28.1 Scenario

You receive a dataset named `student_extunderscore_performance.csv`. You are told only:

“Find useful patterns in student performance and tell us what should be investigated next.”

No instructions are given about which plots to make.

28.2 Mission rules

Do not begin by plotting every column. First create a data dictionary. Determine row grain. Identify identifiers. Profile quality. Then form at least five questions.

28.3 Required deliverables

1. A reproducible notebook.
2. A data-quality table.
3. At least one distribution analysis.
4. At least one bivariate relationship analysis.
5. One subgroup comparison.
6. One missingness investigation.
7. One outlier investigation.
8. A documented cleaning decision log.
9. At least two engineered features with justification.
10. A one-page executive summary.
11. Limitations and unanswered questions.

28.4 Scoring rubric

Area	Marks	What earns the marks
Problem understanding	10	Clear question, row grain, data source and assumptions.
Python/Pandas practice	20	Correct, readable, reproducible code.
Data quality	15	Missing, duplicates, types and inconsistencies investigated.
Statistics	15	Appropriate summaries with interpretation.
Visualisation	15	Correct plot choice, readable labels, useful interpretation.
Reasoning	15	Evidence-based conclusions and limitations.
Communication	10	Clear report, recommendations and next steps.

28.5 Self-review before submission

Ask yourself:

1. Could another student reproduce my results?
2. Did I inspect the data before cleaning it?
3. Did I explain every deletion or transformation?
4. Did I accidentally use future or target information?
5. Do my plots answer questions, or are they decoration?
6. Did I distinguish observation from interpretation?
7. Did I state uncertainty and limitations?
8. Could I explain my project without opening the notebook?

29

Personal Improvement System for the Semester

29.1 The error journal

After each practical session, record three things:

What confused me?	What evidence fixed it?	What will I try next time?
Example: indexing felt random	compared loc and iloc on a tiny DataFrame	predict selected rows before executing

29.2 Weekly 30-minute improvement loop

1. 5 minutes: recall one concept without notes.
2. 10 minutes: solve one tiny coding problem from scratch.
3. 5 minutes: debug one deliberately broken snippet.
4. 5 minutes: explain one chart in plain English.
5. 5 minutes: write one question you still cannot answer.

Think Like an Engineer

The objective is not to become fast by memorising more commands. The objective is to reduce the number of things you need to memorise because you understand the underlying pattern.

Practical Dataset Ladder

Use datasets in increasing difficulty. The links are intentionally clickable so students can obtain the data themselves.

Dataset	Level	Best for	Source
Iris	Beginner	columns, types, grouping, scatter, pair plots	UCI Iris
Titanic	Beginner+	missingness, categorical data, group comparisons	Kaggle Titanic
Wine Quality	Intermediate	distributions, outliers, correlations, scaling	UCI Wine Quality
Adult Census Income	Intermediate	mixed types, encoding, missingness, feature selection	UCI Adult
Bike Sharing	Intermediate+	datetime, seasonality, aggregation, relationships	UCI Bike Sharing
Online Retail	Advanced	transactions, customer cohorts, time analysis	UCI Online Retail

Industry Practice

For healthcare or sensitive domains, use synthetic or officially de-identified educational data. Do not upload real patient, student, financial-account or other sensitive records to a public repository or an unapproved tool. The analytical skill can be practised without exposing people.

References and Official Learning Resources

Bibliography

- [1] John W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977.
- [2] NIST/SEMATECH, *e-Handbook of Statistical Methods*, Exploratory Data Analysis chapters. [Official handbook](#).
- [3] pandas development team, *pandas User Guide*. [Official documentation](#).
- [4] NumPy Developers, *NumPy Documentation*. [Official documentation](#).
- [5] Michael Waskom, *seaborn: statistical data visualization*. [Official documentation](#).
- [6] Scikit-learn Developers, *Scikit-learn User Guide*. [Official documentation](#).
- [7] Scikit-learn Developers, *PCA API reference*. [Official documentation](#).
- [8] YData, *YData Profiling Documentation*. [Official documentation](#).
- [9] Streamlit, *Streamlit Documentation*. [Official documentation](#).
- [10] SciPy Developers, *SciPy Statistical Functions*. [Official documentation](#).
- [11] UCI Machine Learning Repository. [Official repository](#).

Closing Note

The purpose of this book is not to make a student faster at writing `df.describe()`. It is to build the habit of asking better questions, making assumptions visible, testing ideas with evidence, and leaving behind analysis that another person can reproduce and challenge. Those habits transfer directly to machine learning, data engineering, analytics, experimentation and production systems.