SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)
Department of Mathematics and Computer Science

LECTURE NOTES — BOOK 2

Principles of Data Wrangling

Source text: Rattenbury, Hellerstein, Heer, Kandel & Carreras — Principles of Data Wrangling (O'Reilly, 2017)

UDSC-402: Data Analysis and Visualization	Instructor: Dr. D Bhanu Prakash	Date: 10-08-2026

THE SPINE OF THE BOOK

Data moves through 3 STAGES (raw → refined → production). At each stage you perform 9 ACTIONS. Every action contains WRANGLING. Wrangling is a loop of TRANSFORM ⇄ PROFILE. Transformations come in 3 TYPES (structuring, enriching, cleansing); profiling in 2 TYPES (individual-value, set-based). Data quality is described by 5 CHARACTERISTICS (structure, granularity, accuracy, temporality, scope). 4 ROLES do the work using 3 families of TOOLS.

3 stages · 9 actions · 3 transformation types · 2 profiling types · 5 characteristics · 4 roles

1. Why the Field Exists (Foreword)

The historical shift. Through the late 20th century, data was a **bottom-line accounting medium** — balance the books, obey the rules, roll numbers up to executives. A small IT group engineered one authoritative *golden master*. Governing attitude: *garbage in, garbage out* — only carefully engineered data was thought useful.

The modern posture. Data is now a **top-line value-generation medium** — evidence and raw material for new products, processes and efficiencies. Analysts sit across departments, not inside IT. New slogan: *extract signal from the noise* — agile experimentation over large, diverse, imperfect sources.

Both coexist. Accountancy still needs small, slow-changing curated datasets. But the exploratory fraction of organisational data grows exponentially, and that growth forces IT to rethink its traditional tasks.

50–80% of an analyst's time is spent wrangling data, not running algorithms. The common misperception is that analysis *is* statistics on a fast engine; in practice statistics is the last short step of a long process.

Wrangling carries professional judgement, not just plumbing. It encompasses: knowing what data is available; choosing *which* data and at *what level of detail*; working out how to

meaningfully combine sources; deciding how to distill results to a size and shape that supports downstream analysis. These decisions embody scientific and creative intuition and differ per use case and per source.

2. Chapter 1 — Introduction

2.1 Audience

- Those who manage the analysis/application of data **indirectly** - team managers, directors of data projects.
- Those who work with data **directly** - analysts, engineers, architects, statisticians, scientists.

2.2 The two kinds of value — the temporal axis

	Near-term value	Long-term value
Nature	A known list of questions to answer	Optimistic, exploratory potential
Vague example	"Are people shifting to mobile?" 🥺	"Can we forecast seasonal changes?"
Specific example	"*When* will most customer interactions originate on mobile?"	"What supply-chain risk comes from weather or geopolitics?"
Who does it	Analysts	Data scientists
Risk profile	Low	Months long; **non-trivial risk of negative or ambiguous results**

2.3 The single binding constraint: TIME

Ask people what stops them answering their questions and the answer is **time**. They know the questions and know how to answer them — they cannot wrangle the data into shape fast enough. Both value types are gated by *your* limited time and your *organisation's* limited time.

> **Thesis of the book:** Improving wrangling **creates the time** that unlocks both near-term and long-term value.

2.4 Structure of the book

- **Ch 1–3:** the workflow framework and where wrangling sits in it. Building blocks: data flow, wrangling activities, roles, responsibilities. Goal — coordinate effort *across* projects (wrangling constructive, not redundant or conflicting) and *within* a project (standard language and operations → productivity and consistency).
- **Ch 4–7:** the nitty-gritty transformations — a rough *how-to*, explicitly **not** a comprehensive tutorial.

- **Ch 8–9:** roles and responsibilities, then tools.
- Running example throughout: the public **US campaign-finance (FEC)** dataset.

2.5 Case study — Magic thresholds, PYMK and growth at Facebook

Context. By late 2015 Facebook reported >1 billion daily active users with ~17% YoY growth.

The growth equation: **active users = new users + returning users + resurrected users**. *Returning* = active from one period to the next. *Resurrected* = went inactive, then re-engaged.

Core value proposition (Alex Schultz, VP Growth): connecting people to content **from their friends**. So users need friends — but not just any friends, since engagement varies enormously by content.

The near-term questions generated:

1. How many friends does a new user need to be X% likely to return in 30 / 60 / 180 days?
2. What characteristics of a new user's friends distinguish those who churn from those who stay?
3. Do these findings vary by **cohort** (users who joined at around the same time)?

The finding — a "magic threshold": *new users should connect to 10 friends within 14 days.*

Definition — magic threshold: A magic threshold has two properties:
(1) it is a concise **KPI target that predicts** — and with luck drives — the impact you want;
(2) it is **actionable**. KPI targets are ubiquitous; what distinguishes a magic threshold is that it **exposes the core dynamic of the system and gives you a lever**. This one also encodes the core value proposition and coordinates many product decisions behind a single number.

Delivering on the threshold. *Short-term / manual:* let users import an email contact list, triangulated against known users. *Long-term / automated:* **PYMK — People You May Know**, a recommender in the family of Amazon's product and Netflix's title recommendations. It counts as long-term value because the analysis and experimentation take months-to-years, and because the output is a **data-driven service performing the operation automatically**.

How PYMK works. It exploits a UX rule: **recognition is better than recall** — saying yes/no to suggestions is easier and more enjoyable than generating candidates by search or menu. Mechanically: collect friends-of-friends the user is not yet connected to, rank by features such as number of mutual friends, age similarity, education similarity, then present as recommendations.

The counter-intuitive twist: Facebook found PYMK **more effective when aimed at heavy, long-tenured users with vast, diverse networks** than at new users. Bootstrapping a new user is hard and their early recommendations carry low-confidence scores. The winning move is to **recommend the new user to the heavy user** — priming the new user with interesting content, while the heavy user's network yields better friend estimates directed back at the new user.

The generalisable pattern: clear motivation → explicit near-term questions yielding business-improving insight → over the long term, those insights blossom into **data services that automate and optimise** the earlier insights.

3. Chapter 2 — A Data Workflow Framework

3.1 The second axis of value: how it is delivered

	Indirect value	Direct value
Mechanism	Influences people's decisions or inspires process change	Feeds automated systems that act without human input
Canonical example	Risk modelling in insurance	Netflix's recommendation system
Others	Accounting; experimental design in medical research; intelligence analytics; reports; interactive visualisations	High-frequency trading (routing money); Walmart/Amazon (routing goods); Netflix/Comcast (routing content); antilock brakes (routing energy to wheels); the GRE sequencing questions dynamically

Why reports deliver value indirectly: the viewer incorporates the presented information into their understanding of the world, and their updated understanding improves their actions.

Ordering rule: Indirect, human-mediated value is a **prerequisite** for direct, automated value. Human oversight is needed first to discover what is "in" the data and to judge whether quality is high enough for automated use. You cannot send data blindly into an automated system and expect valuable results.

3.2 The three data stages

Stage	Primary objectives
Raw	Ingest data · data discovery and metadata creation
Refined	Create canonical data for widespread consumption · conduct analyses, modelling, forecasting
Production	Create production-quality data · build regular reporting and automated data products/services

- **Raw stage question set:** What kinds of records are in the data? How are record fields encoded? How does this data relate to the organisation, to its operations, and to data already in use?

- **Refined stage:** remove unusable parts, reshape poorly formatted elements, establish relationships across datasets; assess quality issues, because these will damage any downstream automation.
- **Production stage:** feed automated products/services, or enter established pipelines for regular reporting.

Where projects end: A minority end in raw or production; the **majority end in the refined stage**, adding indirect value through insights and models. Canonical example: **Google's Project Oxygen** — a multi-year study of what makes a good manager and whether those characteristics can be taught. Results influenced employee behaviour indirectly; the study data never entered a production pipeline.

3.3 The traditional IT ↔ line-of-business hand-off, and why it breaks

Traditionally the hand-off happens **in the refined stage**. IT owns **ETL (Extract–Transform–Load)**, moving data through the stages under central control; lines of business own analysis — reporting, ad hoc research, advanced modelling, data-driven operational change.

Intended benefits: (1) basic **data governance** from centralised processing; (2) **efficiency** from IT engineers reusing broadly useful transformations. *Actual outcome:* those benefits are often eclipsed by organisational inefficiency and bottlenecks from analysts' dependence on IT.

The two primary bottlenecks

1. **Time to wrangle.** Even starting from refined data, non-trivial transformations remain — removing unnecessary records, joining in extra information, aggregating, pivoting.
2. **Capacity mismatch.** A large pool of analysts waiting on a small pool of IT professionals. This is an *organisational* problem; the fix is widening access to raw data plus the requisite training and skills.

3.4 The data value funnel

Every production use of data depends on **hundreds or thousands** of exploratory, ad hoc analyses. So there is a funnel — many **data sources** → many **exploratory analyses** → a few units of **value** — and, as with any funnel, conversion is nowhere near 100%. Therefore you want as many people as possible exploring data to discover a relatively small number of valuable applications.

Two critical points:

1. Data can produce insights that are **not useful** — not actionable, or too small to justify changing an existing process. *Mitigation:* empower the people who know your business priorities to explore the data themselves.

2. Exploratory analytics must be **as efficient as possible**. Faster wrangling → more explorations → more analyses movable into production. An effective wrangling workflow lets **more analysts explore more data, faster**.

3.5 The holistic workflow framework — nine actions

Nine actions, three per stage. In each stage there is **one primary action whose output is data itself**, and **two output actions whose results are derived from the data** (insights, reports, products, services).

Stage	Primary action (output = data)	Output actions (output = inference)
Raw	Ingest Data	Describe Data · Assess Data Utility
Refined	Design & Refine Data	Generate Ad-Hoc Reports · Prototype Modeling
Production	Optimize Data	Regular Reporting · Data Products & Services

Note: links are drawn one-directionally for simplicity, but **real projects loop back and iterate**. Organisations customise the framework (variants per customer, or multiple refined/production locations where security requires business units to be walled off), but **most use the uncustomised version**.

3.6 Raw stage actions

Three primary actions: **ingestion of data**, **creation of generic metadata**, **creation of proprietary metadata**. Ingestion outputs *data*; the metadata actions output *insight derived from data*.

(a) Ingesting known and unknown data — the complexity spectrum

Complexity	Mechanism	Trade-off
Low	Files by email, shared network folders, FTP	Simple; manual
Medium	Proprietary platforms — **Alteryx, Talend, Informatica Cloud**	Broad transfer/ingestion functionality; eased configuration and maintenance for non-engineers
High	Open-source **Sqoop, Flume, Kafka**	Granular, real-time transfer; requires non-trivial software engineering to set up and maintain

Traditional enterprise warehouses apply some transformation during ingestion — mainly **mapping inbound elements to standard warehouse representations** (each CSV field → a particular relational column). Data is stored in predefined locations, usually **appending** to related

prior data. Appends are simple when new records go at the end; they become complicated when incoming data contains **edits to prior data**, which typically forces ingestion into a **separate location** where more complex merge rules are applied in the refined stage.

Relaxing the constraints: NoSQL (MongoDB, Cassandra) imposes less rigid syntax constraints while keeping many classic access controls. **Distributed filesystems** (HDFS, Amazon S3) look and act like regular filesystems — folders and files with per-file/per-user access control; ingestion can be as simple as copying files or storing a stream into files.

Definitional — memorise: **Schema-on-read** — you do not construct or enforce usable structure until you need to *use* the data; the work is deferred to the refined stage (HDFS/S3 style). **Schema-on-write** — data must adhere to structural and syntactic constraints *in order to be ingested* (traditional warehouse style). The two ends differ only in **when initial enforcement of structure happens**. **Critical caveat:** across the whole spectrum you *still* need a separate refined stage, because refined data has been further wrangled to align with foreseeable analyses.

(b) Creating metadata

- **Generic metadata** = *describing* your data — its characteristics. Broadly useful to anyone working with the dataset.
- **Custom / proprietary metadata** = using those characteristics to judge the data's **value** for a specific analysis or organisation. It **builds on or extends** generic metadata.

Both are composed of the **same five base characteristics**. Metadata actions are triggered when data is **partially or completely unknown** (e.g. a new source).

Vocabulary reset: Datasets are composed of **records**; records are composed of **fields**. Records represent people, objects, relationships or events; fields represent measurable aspects of them. **Records ≡ rows; fields ≡ columns.**

3.7 THE FIVE CHARACTERISTICS

These are the key aspects of **representational consistency** — and therefore precisely the things your wrangling must tune or improve. They reappear in Ch 3 as the *targets* of transformation, and again in Ch 2.8 as the **boundary of permissible variation** in production.

(1) Structure — the format and encoding of records and fields

- **Rectangular** — fixed rows × columns; rows = records, columns = fields.
- **Jagged** — record fields not consistent (extra or missing fields); no longer perfectly rectangular. Supported by **JSON** and **XML**.
- **Heterogeneous** — a mixed set of record *types* (a retail dataset mixing customer information with customer transactions). Common in multi-tab Excel workbooks. Most tools require these to be **split into separate files/tables**; analyses needing all of them start by **joining or blending on shared fields**.

Encoding checklist: timestamps local or UTC? Do date/time fields conform to **ISO 8601**? ages in years, days or fractions of decades? balances in USD? postal codes up to date? are **accented characters** supported or stripped? are **right-to-left** writing systems supported?

Plain text vs binary: plain text is often advisable because it is **human-readable**; the drawback is **file size** — binary encodings, or compression with **gzip/bzip**, are far more space-efficient.

Generic vs custom: assessing structure is **primarily generic** (count records and fields, determine encoding). It becomes **custom** when specific to your situation — a CPG analyst determining which extra fields each trading partner adds, where Walmart's field set differs slightly from Target's.

Assessment questions:

1. Do all records contain the same fields?
2. How do you access the same field across records — by position or by name?
3. How are records delimited? How are fields delimited?
4. How are values encoded (human-readable strings, binary numbers, hash keys, compressed, enumerated codes)?
5. What is the **complexity** of the encoding — primitives or higher-order elements (key-value sets, arrays)?
6. What are the **semantics**, and do they entail quality checks?
7. Are field relationships **singular** (exactly one date of birth) or **set-based** (many shipping addresses)?

(2) Granularity — what one record represents

Described as **coarse** vs **fine**: the level of depth, or the number of distinct entities represented by a single record.

- Fine: one record = one sales transaction, by one customer, at one store.
- Coarser: one record = total sales per store per day.
- Coarser still: one record = total sales by region and week.

Whether granularity is *right*, *too coarse* or *too fine* depends entirely on intended use.

> **The subtlety (custom metadata):** Records that appear to represent **customers** may in fact represent all known **contacts** (only a subset of whom pay). That contacts dataset may hold **multiple entries for the same person** who signed up through multiple channels (Facebook *and* a direct website visit). The honest description of the granularity may then be **"registration events."**

Assessment questions: What kind of thing (person, object, relationship, event) does a record represent? Are records homogeneous or heterogeneous? What **alternative interpretations** exist?

(3) Accuracy — the quality of values

Worked example: a customer-actions dataset of add-to-cart events where the item reference is inaccurate — a **UPC with missing digits**, or an **out-of-date UPC since reused for a different item**.

Other common inaccuracies: misspellings of categorical variables (street or company names); **lack of appropriate categories** (ethnicity labels that cannot express multi-ethnic people); numerical **underflow and overflow**; **missing field components** (a 12-hour timestamp with no AM/PM indicator).

Frequency outliers. Accuracy also covers values occurring more or less often than expected. Judging this is **primarily a custom-metadata concern**, because plausibility relies on organisational knowledge — e.g. an analyst who knows a specific trading partner only ever reports UPCs within a particular range.

Type-specific assessment questions: *Date-times* — time zones included or normalised to UTC? AM/PM demarcated? month/day positions ambiguous (01/02/17 vs 02/01/17)? **an unusual mass of birth dates on the first year of a decade, in January, or on the first of the month?** *Addresses* — components consistent? Is the postal code correct? GPS matching? *Numeric items* (phone numbers, UPCs) — missing digits? invalid subcomponents such as an invalid area code? *Names* — misspellings? missing components? *Email* — valid domain? *Sales transactions* — all amounts in the same currency? signs of fraud?

Mechanism-related questions: Could **sensor drift** have caused systematic inaccuracy over time? Is the data **entered by people** (expect misspellings and non-standard abbreviations)? What is the measurable **distribution** of inaccuracies — do they affect many records, and are they **concentrated in a particular subset**?

(4) Temporality — a record represents an entity at a time

Even a dataset accurate and consistent when created can be rendered inaccurate by subsequent change. *Worked example:* use customer actions to determine the distribution of items people own; weeks later some items are returned. The dataset remains an accurate representation of the **original sales transaction** but is no longer accurate about **what a person owns**.

Explicit vs implicit time. Events usually carry timestamps; non-event records (a customer database of names/addresses/demographics, a reference dataset of UPCs) usually do not. Even so you must understand how time may have affected them — a customer **moves** and their address goes stale; **UPC numbers are recycled** and item descriptions become inaccurate. Knowing **when the dataset was generated** predicts the likely inaccuracies.

Assessment questions: When was the dataset collected? Were all records/fields collected at the **same time**, and is the temporal range significant? Are collection timestamps available as a field or as metadata? Have values been **modified after creation**, and are the modification timestamps available? How can you tell if the data is **stale** (sample and verify manually, or verify via

third-party services)? If there are **conflicting values**, can timestamps determine which is correct? Can you **forecast when** values will go stale?

(5) Scope — breadth of attributes and coverage of the population

1. **Number of distinct attributes** — each attribute is generally one field. **Wide scope** = many fields; **narrow scope** = few.
2. **Population coverage, attribute by attribute** — are all members represented, or have some been excluded **randomly, intentionally, or systematically**?

On breadth: more attributes extend analytical potential, but — as with granularity — **include only as much detail as you might use, and no more**. The right level depends on methodology: **deep learning** favours keeping many redundant attributes and letting statistical methods boil them down; other methodologies operate best with a limited number.

On coverage: usually **not all entities are represented**. Causes include external factors (logging fails in a power outage) and operational error (a chunk of file destroyed, a sensor knocked off power, an incomplete file sent, a redaction). **Knowing the cause matters because it lets you account for the bias downstream** — if under-18s were scrubbed for legal reasons, expect a higher average age than the true population.

The sampling idea: Distinguish the ideal "true" source of data in the real world from the **imperfect record actually acquired**. Statisticians' **sampling** means acquiring a small but representative subset as a stored dataset. The ideal source is often infeasible to capture — consider the temperature of every cubic inch of air over the globe every second; we sample it with thermometers. **Recorded data is inherently a sample.** This is a major difference between real-world (IoT) data and electronic data such as transactions or click logs.

Systematic bias invalidates inference. Canonical example: **drug-trial analysis**, where granularity is patients in the trial. If scope has been intentionally reduced by systematically removing patients (because they **died during the trial**, or began showing **abnormal biometrics**), any analysis will likely misrepresent the drug's actual impact.

The most common custom-metadata question: **"How can this dataset blend (join or union) with my other datasets?"** The new dataset may be a **disjoint extension** (records for population members missing from your data), an **overlapping extension** (creating a need to deduplicate or harmonise), or a source of **new fields** (household disposable income matched to your customer list).

Assessment questions: Which characteristics are captured by the fields — and which are **not**? Are fields **internally consistent** (does age match date of birth; do purchased items sum to the transaction total)? Can you **deduce additional characteristics** from those you have? Are the same fields available for **all** records via the same specification? Do the records represent the **entire population**, and are missing records **random or systematic**? Are there **multiple records for the same thing** — does that change granularity or require deduplication? Is the record set **heterogeneous**?

3.8 Refined stage actions

Three primary actions: **design and preparation of refined data**, **ad hoc reporting**, **exploratory modelling and forecasting**.

(a) Designing refined data

Overarching goal: *simplify the foreseeable analyses you want to perform.* You will not foresee every analysis — initial insights inspire new directions, requiring new or modified refined datasets. **You iterate within the refined stage.**

Contrast with ingestion: raw ingestion involves *minimal* transformation, just enough to satisfy the storage system's syntactic constraints. Designing refined data involves a **significant amount** of transformation, guided by (a) the range of planned analyses and (b) the metadata generated in the raw stage. **Any quality issue identified against the five characteristics should be remedied here.**

Characteristic	Refined-stage remedy
Structure	Most tools expect **tabular** data — every record with the same fields in the same order. For modelling you may additionally convert **categorical data into separate indicator fields** (a gender field → "is male" / "is female").
Granularity	Prudent rule: **build at the finest resolution of record you plan to analyse.** But if most analyses work at a coarser grain (segments, demographic groups, lifetime totals), also store a coarser version — **retaining multiple versions at different granularities streamlines group-based analyses.**
Accuracy	Three strategies: **(a) Remove** records with inaccurate values (provided they are detectable); **(b) Retain but mark** them as inaccurate, still allowing some analyses; **(c) Replace** with default or estimated values — **imputation** (e.g. cap extremely large transaction amounts at a maximum so averages and standard deviations are not biased).
Temporality	Resolve conflicts with an explicit reference to time — **assign date ranges** to multiple addresses; keep a transaction that violates *current* business logic if it occurred **before that logic was enforced**. Distinguish the time an action **occurred** from the time it was **recognised** (common with bank transactions); sometimes a **version number** beats a timestamp (software version, data-file version). **Preserve and document all timestamps and version numbers available.**
Scope	Design refined datasets to include the full required set of records and fields; **build a fully blended dataset** where most analyses need contact information plus transaction summaries. The **most important scope challenge is understood population coverage** — ideally one and only one record per population member. A **true random subset** supports valid inference about the population; a **biased** subset restricts what you can validly conclude.

Conflicting information — between fields (multiple addresses; deviation between date-of-birth and age) or between a field and business logic (an impossibly large transaction) — is the primary accuracy issue here. When the affected fraction is small, **removal** is appropriate; otherwise **reconcile** (e.g. recompute age from date of birth and the date of the events being analysed).

(b) Ad hoc reporting

The core action for answering **specific questions**; BI analytics and dashboarding are specific forms. These analyses are **primarily retrospective**. Output ranges from a single statistic to a full report with discussion. Because the initial query constrains consumption, there is unlikely to be an automated consumer — the value is **indirect**.

Range: from succinct definitive questions ("how many customers purchased item X last week") to **open-ended investigations lasting months or years** ("identify the key factors driving the customer shift from desktop to mobile").

The classic trigger — an anomaly in a regular report. The assessment sequence:

1. Is there a **data reporting or data quality problem**?
2. If the data is valid — i.e. the anomaly reflects a change in the *world*, not merely in the *representation* of the world — can it be **isolated to a subpopulation**?
3. What other changes are **correlated** with it?
4. Are those changes linked by **causal dependencies** to one another, or to a **common root change**?

(c) Exploratory modelling and forecasting

Unlike ad hoc reporting these are **primarily concerned with the future**: *what do we expect to happen given what we have observed?* In **forecasting** the explicit objective is to predict future events — next quarter's sales, next month's churn percentage, each customer's renewal likelihood. Predictions are usually built on **models**; for some analyses the useful output is **not a prediction but the model itself**.

Why model: to understand the **relevant factors driving a behaviour**. Modelling is a common prerequisite for distinguishing **correlated** from **causal** factors, but **causal analysis also requires carefully designed experimentation** — holding some factors constant while perturbing others, and/or modifying the underlying system to decouple or realign factors.

Back to Facebook (the cleanest illustration in the book): Simple analysis of friend count vs long-term retention shows it is **not** causal — there are long-tenured users with few friends, and heavily connected users who churned. The factors are nonetheless **strongly correlated**, and the

product changes Facebook made to increase friend counts **demonstrably move the underlying causal factors** that do drive retention. Correlation can be actionable without being causal.

3.9 Production stage actions

Once refined data generates valuable insights, you separate analyses that must be **regularly refreshed** from one-offs. Exploring and prototyping is one thing; **wrapping those outputs in a robust, maintainable framework that can automatically direct people and resources is another game entirely.** Trigger statements: *"We should track that measure all the time"*; *"We can use those predictions to expedite shipping of certain orders."*

(a) Creating optimised data

	Refined data	Optimised data
Intended use	Broad, partly unforeseen	**Highly specified**
Goal	Support the **widest** set of analyses as efficiently as possible	Support a **very narrow** set **robustly and efficiently**
Extra specifications	Scope, accuracy of values	Plus **processing and storage resource constraints**, which dictate the structure of the data and how it is made available to the production system

(b) Designing regular reports and automated products/services

More than wiring data into report logic. The major additional work is **monitoring the flow of data** and ensuring **structural, temporal, scoping and accuracy constraints remain satisfied over time**. Those four constraint families **define the boundary of permissible variation** (minimum and maximum sales amounts; coordination between billing address and transaction currency). **Within** those bounds the logic must handle variation — a real departure from exploratory analytics, which may legitimately use dataset-specific logic. **Production logic must be generalised.**

Common dataset variations that force wrangling-logic changes: extensions to **value ranges** (current dates; redefined regions or customer segments); **new accuracy issues** (previously unseen misspellings); fields **removed or emptied** (age or gender redacted for compliance); appearance of **duplicate records**; **disappearance of a subset** of records (a change in segment names drops one or more groups). You may instead **tighten the boundary** to exclude duplicates or missing subsets — in which case the catching-and-remedying logic lives in the **data optimisation** action.

3.10 Where wrangling sits

Definition: *Data wrangling is the process involved in transforming or preparing data for analysis.* In the framework it **occurs between the stages** — the set of actions that move you from raw → refined, and from refined → optimised/production.

Relationship to ETL: in the raw→refined transition, wrangling can resemble traditional ETL. **ETL is* one type *of wrangling** — specifically the type managed and overseen by shared services or IT. But wrangling can equally be done by business users in Excel, or by data scientists in Python or R.

Key claim: wrangling is a **core task within every action** of the framework — but the tasks are **not identical** across actions; they differ especially in the *kinds of transformation* applied.

4. Chapter 3 — The Dynamics of Data Wrangling

4.1 The four steps: Access → Transform ⇄ Profile → Publish

Step	What it involves
Access	Wrangling begins here, sometimes gated on permissions and infrastructure changes. Also involves **manipulating locations and relationships between datasets** — moving datasets around a folder hierarchy, replicating them across warehouses for easier access, analysing differences between similar datasets and assessing overlaps and conflicts.
Transform	The bulk of the work: manipulating the **structure, granularity, accuracy, temporality and scope** of the data to better align with your analysis goals. *(Note how the five characteristics reappear as the targets of transformation.)*
Profile	The **meaningful feedback** that assures the manipulator a manipulation succeeded. A predefined, somewhat generic set of feedback is often sufficient; sometimes **customised profiling** is required. **The bulk of wrangling is frequent iteration between transforming and profiling.**
Publish	Understood by *what* is published — (1) a **transformed version of the input datasets**; (2) the **transformation logic itself** (a script generating a regular report); (3) **profiling metadata** about the dataset — *these profiling reports are critical for managing automated data services and products.*

The loops. Iteration is not only transform ⇄ profile; there are less frequent loops **back to access**, and during or soon after publishing you may realise the output isn't right and need further transformation or profiling.

4.2 Two efficiency levers

(a) Subsetting — for heterogeneous data

When a dataset holds a heterogeneous set of records — differing in **structure** (extra or different fields) or in **granularity** (some records are customers, others accounts) — the best approach is to **split the dataset, wrangle each subset separately, then merge results back if needed**.

The underlying principle: **Process efficiencies are fundamentally rooted in applying the same processing mechanism to many similar inputs.** At one scale that means transformation steps that operate across every record; at a wider scale, one script that operates across multiple similarly structured datasets.

(b) Sampling — for big data

Applies when the dataset is too large to review record by record, or so large that even simple transformations take prohibitively long. The pain is felt in the **feedback loop**: change a derived calculation or grouping rule, then wait a minute, ten minutes, or half a day. Wrangling then **dominates your workflow and you get through a few analyses**.

The fix — and the number to quote: Work with **samples you can transform and profile at interactive time scales** — ideally within **100 milliseconds**, occasionally up to a few seconds. The connection to profiling is direct: deliver profiling feedback from a sample rather than the whole dataset. (Statistical subtleties: Kish, *Survey Sampling*; extraction techniques: Cormode et al., *Synopses for Massive Data*.)

Working example — why sample design matters. You derive customer tenure from the registration date. You already have expectations: the business started 11 years ago, so **no customer should show a duration over 11 years**; there was a big customer increase about 3 years ago, so you expect a **bump around 3**. To check these you need a sample containing **extreme values** (earliest and latest registration dates) *and* a **random sample** over the rest so overall distributional trends are visible.

Worked example — stratified sampling. Case-based transformations by record group: converting transaction amounts to USD when the data contains Euros, GB Pounds and so on, each requiring its own conversion. To confirm every conversion applied correctly you must profile results **covering all currencies present**.

Definition — stratified sample: A sample covering all groups ("strata"), providing representation of each group even though it may **bias the overall trends** of the full dataset by over-representing small groups relative to large ones.

4.3 The three core transformation types

Type	Definition	Detail
Structuring	Actions that change the **form or schema** of your data	Primarily moves record field values around, sometimes summarising them. Simplest: change the **order** of fields. Per-record: **break fields into components** or **combine fields into complex structures**. Inter-record: **remove subsets of records**. Most complex: **aggregations and pivots** — aggregations shift granularity (customers → segments; transactions → quarterly net revenue); pivoting shifts records into fields or fields into records.
Enriching	Actions that **add new values** to your dataset	Typically from **multiple datasets**. Three classes: (1) **joins and unions**; (2) **inserting metadata**, dataset-independent (current time, username) or dataset-specific (filenames, record locations); (3) **computing new values** — generic derivations (time conversion, lat/long from an address, a sentiment score from a support chat log) or custom (mineral-deposit volumes from rock samples; health outcomes from treatment records).
Cleansing	Actions that **fix irregularities** in your dataset	Fixes quality and consistency issues **predominantly by manipulating individual field values within records**; the most common variant fixes missing/NULL values.

Erratum worth knowing: The printed text swaps the definitions of *join* and *union* in the Chapter 3 passage. Chapter 6 gives the correct ones — **unions stack vertically, joins link horizontally**. Use the Chapter 6 definitions in the exam.

4.4 The two core profiling types

Type	Definition
Individual-values profiling	Understanding the **validity of individual record fields**. Two constraint families — **syntactic** (formatting; a date should be `MM-DD-YYYY`) and **semantic** (context or proprietary business logic; your company is closed on New Year's Day, so no transactions should exist on January 1 of any year). Ultimately determines the validity of individual field values, or by extension of **entire records**.
Set-based profiling	Understanding the **distribution of values** for a given field across multiple records — the **shape and extent** of a distribution, or

Type	Definition
	relationships **between multiple fields**. Example: you expect retail sales to be higher in holiday months, so you build a set-based profile confirming sales are distributed across months as expected.

4.5 Second-order concerns — doing the work well

- **Optimise the script** to run efficiently and robustly;
- **Track changes** to it over time as new data mandates edits or better formulations appear;
- Possibly **manage multiple versions** for legacy and capability purposes;
- Support all of the above with **additional customised profiling**, tracking statistics **across variations of the dataset**.

4.6 Wrangling inside each of the nine actions

Action	What the wrangling looks like
1. Ingesting data	Non-trivial transformation may be needed just to conform to basic structural requirements; the extent depends on infrastructure (older warehouses demand particular formats; MongoDB or HDFS permit more variety). **Explicit goal: minimal transformation needed to make the data available for metadata analysis and eventual refinement.** Governing objectives: ***"don't lose any data"** *and* **"fixing quality problems comes next."** *In practice: limited structuring plus enough profiling** to ensure nothing was lost or corrupted.
2. Describing data	Assessing the five characteristics is **profiling-heavy**, but broad understanding also needs exploratory **structuring**: breaking out sub-components to assess quality; filtering to subsets to assess scope and accuracy; aggregating and pivoting to triangulate values against internal and external references.
3. Assessing data utility	Primarily **enriching and cleaning**. If the dataset is a new installment you must assess the ability to **union** it; you will also **join** it to existing data — and the attempt reveals linking problems: **too few links**, or equally problematic **too many duplicative links**. Treating existing data as the **baseline standard**, you spend real time cleaning the new dataset to tune overlap. **Set-based profiling** gives the basic feedback on blend quality.
4. Designing & building refined data	Requires the **breadth of both transformation and profiling types.** *Structuring* to align with intended granularity and scope. *Enriching* — blended datasets via joins and unions; build non-trivial derivations (smoothed time-series, sentiment scores) into the refined dataset rather than making every analysis rebuild them. *Cleaning* — **at minimum flag** inaccurate values, or replace them; many analyses need missing data filled with reasonable estimates. *Profiling* — apply individual-value

Action	What the wrangling looks like
	profiling **aggressively across all fields**; initially little more than **syntactic** constraints, because semantic constraints are unknown. **As more ad hoc analyses complete, semantic checks get added** and set-based profiling grows richer — from simple range checks to expected correlations and trends (a steadily increasing median sale price).
5. Ad hoc reporting	Primarily **structuring/restructuring** from refined data: pulling sub-components out of fields; filtering out non-relevant records; aggregating or pivoting around subgroups. May also need **enrichment** — much join/union work is already done, but not all — plus derived values such as the current date, arbitrary percentiles, or **winsorized averages**.
6. Exploratory modelling & forecasting	Significant **structuring**. Beyond filtering, aggregating and pivoting, it is common to **pivot categorical fields into separate indicator fields**, which is what enables techniques like **regression**; fields may need **normalising** so model weights are comparable. *Side effect:* modelling **indicates outliers** — either (a) the outlier contains **inaccurate values**, instigating extra cleaning, or (b) the outlier is **valid** and forces a change in how you understand the data. *Robustness analysis:* re-run the model after transforming inputs and deleting records.
7. Building an optimised dataset	Needs the breadth of both, but **the balance differs**: refined data needs a fairly even mix; **optimised data requires significantly more profiling**, because the scripts must be **applied automatically to regularly updating input**. Profiling must (a) **forecast the robustness** of scripts to future data variants and (b) **track correctness** every time they run — mostly by examining **distributions of values for various subsets of records**.
8–9. Regular reporting & data products/services	Relative to their refined-stage counterparts, **similar transformations but more profiling** — driven by the requirement that the **same scripts must work efficiently and robustly across evolving input data.**

4.7 The emphasis map (high-yield table)

Stage	Action	Emphasised transformation	Emphasised profiling
Raw	Ingesting Data	**Structuring**	**Individual value**
Raw	Generating Generic Metadata	**Structuring**	**Individual value + Set-based**
Raw	Generating Proprietary Metadata	**Enriching + Cleaning**	**Set-based**
Refined	Designing & Building Refined Data	**All three**	**Both**

Stage	Action	Emphasised transformation	Emphasised profiling
Refined	Ad-Hoc Reporting	**Structuring**	—
Refined	Exploratory Modeling & Forecasting	**Structuring**	—
Production	Designing & Building Optimized Data	—	**Individual value + Set-based**
Production	Regular Reporting	**Structuring**	**Individual value + Set-based**
Production	Building Products & Services	**Structuring + Cleaning**	**Individual value + Set-based**

Pattern to remember: Structuring is emphasised nearly everywhere; **enriching and cleaning peak at proprietary-metadata generation and refined-data construction**; **profiling weight grows monotonically as you move toward production.**

5. Chapter 4 — Profiling

5.1 Why profiling comes first

It is the **first action people undertake when beginning each stage** of a data project, because **you must understand the contents of your data before you can transform or analyse it**. Fundamentally, **profiling guides data transformations.** Three jobs it does:

1. On a real project you **don't have time to look at every field of every record** — profiling tells you what is "in" the dataset.
2. It **validates that your transformations worked as intended**.
3. It **assesses data quality** and pre-empts transformation failures — if downstream analysis expects a numeric price column, you need to know now about a record containing letters or special characters.

The two views — examining **individual values**, and examining a **summary view across multiple values**. Either can be consumed **textually** (a list of values, a table of summary statistics) or as **visualisations**.

5.2 Individual-value profiling — syntactic

Definition. *Syntax* refers to constraints on the **literal values** valid in a field. The permissible set is often fairly large — e.g. the lifetime count of ATM cash withdrawals by a bank customer, where permissible values are integers from 0 to 50,000, the upper limit derived as *permitted withdrawals per day × days the bank has been in business*.

Note the reasoning: Syntactic bounds are themselves **derived from domain facts**. **Method:** check whether values are in (or not in) the set of permissible values. **Best example-level output:** a **random subset of values that satisfy** the constraint **plus a random subset that do not** — from those you can usually judge whether **a different syntactic type would fit the data better**.

5.3 Individual-value profiling — semantic

Definition. Semantic constraints correspond to the **meaning or interpretation** of field values: values are valid if their **interpretations** satisfy the constraints.

Case	Example	Reading
1. Syntactically invalid, semantically meaningful	Biological age in years, with unreported ages stored as `-1`	Not a valid age; semantically a **Boolean indicator of whether the customer reported their age**. Deriving a "reported age" field may be exactly the variable you need — e.g. to analyse how **willingness to report information predicts overall satisfaction**. *Semantic interpretation can override syntactic constraints.*
2. Semantically clear despite bad syntax	`"San Francisco, CA USA"`, `"Sn Francisco, CA, USA"`, `"San Francicso, CA, USA"`	Some are syntactically invalid but **semantically clear**, and can be converted to their syntactically correct versions.
3. Genuinely ambiguous	The value `"Moscow"`	Moscow, Russia, or Moscow, ID, USA? Other fields may disambiguate; **on their own these values are semantically invalid.**
4. Simple categorical conversion	Age → life stages; time of day → day stages	Teenager / young adult / senior; morning / afternoon / evening / night.

The general method: Profiling semantic constraints often requires **deriving a new record field that explicitly encodes the semantic interpretation** of a source field. That explicit encoding can then be **syntactically typed**, and validity determined by testing whether the value is in the valid set. Hence, as with syntactic types, the most appropriate summary statistics are the **percentage of field values that are valid, invalid, and empty/null**.

Deterministic vs probabilistic mappings. *Common and straightforward:* **deterministic rules** — anyone aged 13–19 inclusive is a teenager. *More difficult:* **non-deterministic / probabilistic mappings** — based on summary demographics, interpret `"Moscow"` as **80% likely** Moscow, Russia and **20% likely** Moscow, ID, USA.

5.4 Set-based profiling by field type

Field type	Profile to build
Numeric	Start from a simple **histogram**; consider comparing against a known distribution such as **Poisson** or **Gaussian**. Alongside it look at **minimum, maximum, mean, sum** — these give the spread and immediately identify problematic distributions or outliers.
Categorical	Two useful profiles: (1) **counts of occurrences of unique values**; (2) **cluster the raw values via a mechanism similar to standardisation, then count values per cluster** — this is what surfaces misspelling families.
Geospatial (ZIP, lat/long)	**Plot on a map.**
Date-time	Plot on a **variety of scales**: across the 24-hour day, the 7 days of the week, the 12 months of the year.
Multiple fields	**Cross-distributional profiling**, commonly with simple **scatterplots** — useful for assessing overall dataset quality.

5.5 The running case study — FEC campaign finance

Project goal: using publicly available US campaign-finance disclosure files, wrangle candidate data and individual-contribution data from the 2016 presidential election to see whether there are **trends in the contributions received by Hillary Clinton and Donald Trump**. Refined-dataset question: *"How were individual campaign contributions distributed between the two 2016 major-party US presidential candidates?"* Profiling is heavy here because the data is largely **unknown**.

Definition — data dictionary: *Data dictionaries* are frequently generated as the output of the **metadata-producing actions during the raw data stage**. They define the permissible valid values in each column and can also explain the contents of the dataset. The FEC supplies one to supplement the raw files.

About the Candidate Master File (2015–16). It contains all candidates who filed the appropriate form for an upcoming election, and **also candidates with active election committees even if the election they registered for is not the current one**. You will notice **creative candidate names** — filing paperwork does not mean the candidate appeared on any state ballot. Two profiling questions for the file: (1) Are there values **syntactically invalid** given the scope of our project? (2) Is the **range of values** in each column **valid** given the scope of our project?

Worked syntactic profiling

- **Column 5 = state of the office sought.** Collect all unique values — **a common first step in syntactic profiling** — producing **57** unique values. Reasoning about what *should* be

permissible: **50 states** with voting representatives, plus **5 territories and the District of Columbia** with non-voting representatives, plus the non-state value **"US"** for presidential candidates → 57 is plausible. Checking each value against a **lookup to a reference dataset of all 57 valid values** (insert 1 where matched, 0 where not) gives **100% valid**.

- **Column 14 = state of the candidate's mailing address.** A count of distinct values again gives 57 — **but only 56 are legitimately possible**, because a mailing address cannot be "US". The column also contains **missing values**, which count as **syntactically invalid because the data dictionary does not indicate missing values should appear**. The same lookup procedure reveals **one record with an erroneous state: `"ZZ"`.**
- *Practice:* perform similar checks on the other columns, **generating a series of Boolean indicator columns** showing whether each record's values are permissible under the data dictionary.

Worked set-based profiling

Column 4 = the election year for which the candidate registered. Because the file includes candidates for **any** election with an active committee, the expected distribution is: few records before 2016, **a large number for 2016**, possibly a few for 2018 and 2020. *Actual:* a summary count per year shows a **very wide range** — earliest **1990**, farthest in the future **2064**.

The correct response — step back and assess utility: Assessing utility means generating **custom metadata** specific to your use case. Since the project concerns only the 2016 presidential election, records for 1990 or 2064 are **irrelevant**. The recommended action is **not deletion** but **inserting additional metadata — flagging records whose column 4 is not "2016" as invalid.**

6. Chapter 5 — Transformation: Structuring

Definition. Structuring involves **changing your dataset's structure or granularity** — any action that changes the **form or schema** of the data.

6.1 The taxonomy

Family	Types
Intra-record (individual records and fields)	(1) **Reordering** record fields (moving columns); (2) **Creating new fields by extracting values**; (3) **Combining multiple fields into a single field**
Inter-record (multiple records/fields at once)	(1) **Filtering** datasets by removing sets of records; (2) **Shifting granularity** and the associated fields, through **aggregations and pivots**

6.2 Intra-record — extracting values

Definition. Extraction creates a **new record field from an existing record field** — frequently by identifying a **substring** in an existing column and placing it into a new column.

(a) Simple positional extraction

Specify the **starting and ending position** of the substring. Most common with **date-time and fixed-width fields**, because both have known elements at specific positions with little structural inconsistency.

Worked example (Individual Contributions, column 14 = transaction date). Values look like `03102015`, `03302015`, `03022015`. Discrete elements always align: month first, day after exactly two characters, year after four. Counting characters **including spaces**, the day of month **starts at position 3 and ends at position 4**. Stated as a sentence: *"From source column 14, extract the characters located from position 3 to position 4."* Result: 10, 30, 02.

Exam trap: The book counts positions from **1**; Python and most languages slice from **0**. Positions 3–4 → Python `[2:4]`. **Purpose of this derived field:** it lets you determine whether contributions cluster at certain times of the month.

(b) Complex positional extraction

Use when the **start and end positions differ from record to record**. You use functions that **search for a particular sequence of characters** within the original field; these return either the **start position** of the sequence or its **length**. You pass those returned values into a basic positional extraction function — producing a **complex nested function**.

Worked example (column 8 = name of the contributor). For individuals a **comma** separates last name from first: `ARNOLD, ROBERT` · `BICKLE, DON` · `ROSSMAN, RICHARD` · `LLEWELLYN, CHARLES`. Last-name lengths differ (ARNOLD = 6, ROSSMAN = 7), so **simple positional extraction would fail**; but the shared **comma delimiter** lets you find the position and then slice. *Also good candidates:* **address fields** and **full-name fields**.

(c) Pattern extraction

Uses **rules describing the sequence of characters** you want, rather than positions or delimiters — typically **regular expressions**, supported in code and by most wrangling software. *Worked example (column 20 = free text describing each contribution):* values like `P/R DEDUCTION ($296.67 MONTHLY)`, `P/R DEDUCTION ($1000.00 MONTHLY)`, ` *EARMARKED CONTRIBUTION: SEE BELOW`. *The sentence is:* "From column 20, extract the first sequence of digits, followed by a period, followed by another sequence of digits." *Use it when you need substrings that conform to the same generic pattern but are not identical*.

(d) Complex structure extraction

Extracting elements from **complex hierarchical structures** — commonly required with **JSON** and other **semi-structured** formats, which often originate from **automated / machine-generated systems**.

	JSON array	JSON map
Represents	An **ordered sequence of values**	A set of **key-value pairs** (key = property name, value = its value)
Enclosed in	**Square brackets**	**Curly brackets**
Separators	Elements separated by commas, enclosed in double quotes	Pairs separated by commas; keys and values both in double quotes
Example	`["Sally","Bob","Alon","Georgia"]`	`{"product":"Trifacta Wrangler","price":"free","category":"wrangling tool"}`

Why they exist: they allow datasets to include a **variable number of records and fields**. *Array variability:* one shopping cart's order array may have two elements and the next three. *Map variability:* each cart might carry any of `gift_wrapped`, `shipping_address`, `billing_address`, `billing_name`, `shipping_name`; **representing them as a map avoids creating a very sparsely populated table.**

The tension: JSON is ideal for **storing data efficiently** but is usually **not structured ideally for analytics tools**, which expect **tabular** input. So you must either **pluck a single element into a new column**, or **fold the multiple elements of an array down into multiple records** — converting JSON into the rectangular structure downstream analytics needs.

(e) When to use each extraction technique

Technique	Use when
Simple positional	Every record conforms to a known structure and each substring **always begins and ends at the same position**. Good candidates: fixed-width fields, date-time fields.
Complex positional	Each field contains multiple substrings separated by the **same delimiter string**; substrings need not be the same length or follow the same pattern. Good candidates: address fields, full-name fields.
Pattern-based	The target substring can be defined by a **generic pattern**.
Complex structure	Fields contain **JSON maps or arrays** and you want individual elements out of them. Common in machine-generated data.

6.3 Intra-record — combining multiple fields

Essentially the **reverse of extraction** — create a single field merging values from multiple related fields. *Worked example:* column 9 (city) adds detail to column 10 (state); combine them separated by a comma → `MCPHERSON, KS`, `FREDERICK, MD`, `BROKEN BOW, NE`, `HAWTHORNE, CA`. **Rationale:** useful when downstream analysis wants to treat the combination as **a single record field**.

6.4 Inter-record — filtering records and fields

Dual role. Filtering is often used in **cleaning** transformations addressing quality (Ch 7), **but** it can also be used as a **structuring** action to alter **granularity**, by changing the *types* of records and fields represented.

Worked example. The Individual Contributions dataset has an entity-type column with **eight distinct values** (`CAN, CCM, COM, IND, ORG, PAC, PTY`…). Because records may belong to any of eight groups, the granularity is **fairly coarse**. Filtering to keep only `IND` (contributions from individuals) produces a dataset of **finer granularity**, because every record now belongs to a **single category**. That is **record-based filtering**. **Field-based filtering** is the other type — it affects the **number of columns**.

6.5 Inter-record — aggregations and pivots

Structuring operations that **shift the granularity** of a dataset. *Simple case:* total sales by week, store or region — straightforward summation. *More complex pivot:* extract items purchased out of transaction records and build a dataset where **each record corresponds to an item**, with fields describing the product and an aggregated count of transactions involving it; or count transactions per product **where the product was purchased alone, with one additional product, with two additional products**, and so on.

Organising principle — three progressively more complex groups, characterised by the mapping between input and output records:

Operation	Input → output mapping	Worked illustration
Simple aggregation	**Each input record maps to one and only one output record**, while **each output record combines one or more input records**. Output fields are simple aggregations — sum, mean, min, list concatenation.	Contributions to campaign committees (col 1 = committee, col 15 = amount). C00004606: 750 + 1000 → Sum 1750, Mean 875, Count 2. C00452383: 225 + 50 → Sum 275, Mean 137.50, Count 2.
Column-to-row pivot ("unpivoting" / "denormalizing")	**Each input record maps to multiple output records**, and **each output record maps to one and only one input record. Input field values become the defining**	Region × {2015, 2016} sales → (Region, Year, Sales) triples: East/2015/2300, East/2016/2453, West/2015/9866,

Operation	Input → output mapping	Worked illustration
	characteristics of the output records; output records contain a **subset of the input fields**.	West/2016/8822, Midwest/2015/2541, Midwest/2016/2575. **Use when** the source has **multiple columns representing the same type of data**; the result **allows you to summarise values more easily** across years and regions.
Row-to-column pivot	**Output records are sourced from multiple input records, and input records may support multiple output records.** Output fields may involve **simple aggregations (sum, max)** or **more complex expansions based on the field values**.	Sum of contributions per committee, broken out by contribution type — one new column per type (Sum of IND, Sum of PAC). A new column is created for each unique value of the source type field, with each row summarising one committee.

Memory aid: **column-to-row = wide → long = unpivot / melt / denormalise; row-to-column = long → wide = pivot**, and row-to-column is also how you make **indicator variables for regression**.

7. Chapter 6 — Transformation: Enriching

Definition. Enriching actions result in the **net addition of information** to your dataset — inserting additional records or fields from other related datasets, or using formulas to calculate new fields. Three primary types: **unions**, **joins**, **deriving new fields**.

The distinction from structuring (a guaranteed exam question): Both can create new fields or records, but **structuring creates them from data already present in the dataset**, whereas **enriching creates them using new data — information not previously present in the dataset in any form.**

7.1 Unions

Definition. Unions **append additional records to an existing dataset** — take two related datasets and **stack them vertically** into one. *Motivating scenario:* your organisation receives monthly orders from clients; at each quarter end you must summarise total orders per client over three months, so the three monthly datasets must be combined first.

- **Simple case:** records from old and new datasets have **matching structures**. The union is **little more than a concatenation**.

- **Complex — and more common:** the structures differ, generally only **minorly**. Simple logic then dictates whether the unmatched fields should be **coerced into the same field**, or **kept separate with null/empty values inserted** into records from the other dataset.

7.2 Joins

Definition. The **most common enrichment action**. The commonest form **links records from one dataset to records from another via exact matches of a single field** in each — the **key field** in both datasets. *Worked example:* Dataset A holds customer profiles keyed on `customer_id`; Dataset B holds transactions with a `customer_id` key. Joining lets you **analyse how transaction activity relates to profile information**. **Fuzzy matching** allows records with **misspellings or minor differences** to be linked; most tools support exact matching on one or more fields, some additionally support fuzzy.

Join type	Behaviour
Inner	Produces a record **only when there are matching records from each dataset**. **If there are duplicate keys, the output records are similarly duplicated.**
Left outer	Retains **all records from the left (initial) dataset**, even where there is no matching record on the right (incoming).
Right outer	Retains **all records from the right (incoming) dataset**, even where there is no match on the left (initial).
Full outer	Retains **all records from both datasets**.

Two failure modes to name in an exam: *Too few links* (keys need cleaning before they match) and *too many duplicative links* (fan-out from duplicate keys, which silently double-counts aggregates). Both are exactly what "assessing data utility" exists to catch.

7.3 Inserting metadata

An important form of enrichment: adding metadata **into** the dataset. Common items: **filenames of the source data, byte offsets and/or record numbers, current date and/or time, creation/update/access timestamps, and record and/or record-field lineage.**

7.4 Derivation of values

(a) Generic derivations — four domains

Domain	What you derive
Time	Most datasets must explicitly address time — encoded **within** records or **across** records as metadata. Common derivations: **day of the week, season**; and when data spans time zones, both a **local and a global (UTC)** timestamp per event. Crucial for **weekly or yearly cycles,**

Domain	What you derive
	experiential time, and absolute sequencing of events (e.g. anomaly detection).
Geography / spatial	From abstracting an address into a **ZIP code or city**, to converting an address to **latitude and longitude** via a service, to converting an address to a **customised marketing or sales region** — drawn, for example, by calculating the **shortest driving distance** between a customer and all nearby stores. **Note: none of the three tools in Ch 9 support spatial encodings — you would need custom code.**
Text / NLP	Derive an overall **sentiment analysis**; **extract references to people, events or things**; **identify topics**; or **translate** between languages. **Key concern:** what defines your **baseline or reference language** — general English, or terminology specific to your industry? That choice determines **how you treat jargon or slang**.
Basic numeric	Simple: summing list price and taxes to a final price. Sophisticated: **aggregation or sequential processing** — e.g. **z-normalising** a field by subtracting its mean and dividing by its standard deviation.

Domain-specificity within "generic". Some generic functionality is domain specific: **region definitions differ** for a mining company versus a CPG retailer; NLP derivations can target jargon specific to **silicon manufacturing, pharmaceuticals, fashion**, and so on.

Legal/compliance-driven derivations. *Healthcare:* patient privacy and confidentiality laws require **certain fields to be removed and replaced with derived ones**; and given a treatment history, you might derive fields indicating **whether certain future treatments are viable**. *Finance:* domain-specific derivations on the **time series of transaction data** — models such as **Black–Scholes** applied to historical trading series to **predict future value**, with predictions then analysed alongside others to **identify the optimal model** or **build a composite prediction**.

(b) Proprietary derivations

Further along the generic-to-specific spectrum. Organisations may have **custom models** for prediction, or **highly customised derivation calculations** capturing, for example, **the health of their customer base** or **the likelihood of a customer leaving or upgrading**.

Implementation — UDFs: In many big-data wrangling systems, proprietary functionality is encoded as **User-Defined Functions**, often supported across several languages — **R, Python, VisualBasic, Java**. While many UDFs contain non-trivial calculations, **others make calls to services** — e.g. Google's geocoding API converting addresses to latitude/longitude.

8. Chapter 7 — Using Transformation to Clean Data

Definition. The third transformation type: cleaning a dataset to **fix quality and consistency issues**, predominantly by **manipulating individual field values within records**. The two most common variants address **missing/NULL values** and **invalid values**.

8.1 Addressing missing / NULL values

- **Filter out** records with missing or NULL fields.
- **Replace** missing or NULL values — often called **data imputation**.

Imputation strategies: insert the **average or median** value; **generate values from similar records** (similar customers or similar transactions); or, if the data has **strong ordering** (a time series), **fill in using the last valid value**.

8.2 Addressing invalid values

Three reasons a value is invalid	Three responses
1. **Inconsistent with other fields** — a customer's age versus their date of birth	**Calculate the correct or consistent value and overwrite** the original
2. **Ambiguous** — two-digit years; an abbreviation like **"CT"**: Connecticut or Court?	**Simply mark values as invalid**
3. **Improperly encoded**	**Run two parallel analyses** — one including the invalid values and one excluding them — **providing insight into the impact that invalid data is having on your insights**

Worth quoting in an exam: In the two-parallel-analyses approach, **the disagreement between the two analyses is itself a measurement of data quality.**

8.3 Standardisation — the more complex variety

Scenario. Every customer is known to reside in the United States, so a reasonable validity check on the state-of-residence field is membership of the set of known US states. But there are misspellings: **"Californa", "Westvirginia", "Dakota"**. **Standardising these to a fixed library of valid values is a good way to improve dataset quality.**

- **Mechanism 1 — edit distance.** The most common method uses **editing distance around misspelling**: strings that are similar, like "Californa" and "California", are treated as the **same entity and converted to the same spelling**.
- **Mechanism 2 — domain knowledge.** Is "Dakota" North or South Dakota? If the record has a **ZIP code** in another field, use a **mapping of ZIP codes to states**. A **slightly less reliable** mapping uses the **area code** on a phone-number field — **less reliable now that cell phone numbers can be transferred across carriers.**

9. Chapter 8 — Roles and Responsibilities

9.1 The four roles and the two axes

Four common job titles: **data engineer, data architect, data scientist, analyst**. In small organisations or personal projects **one person may develop and apply all of these skills**. The roles sit on two axes — **vertical: the skills and methods used** (*technically focused* ↔ *business focused*); **horizontal: the primary kind of output produced** (*internally facing* ↔ *externally facing*).

	Internally facing output	Externally facing output
Technically focused	**Data Engineer**	**Data Scientist**
Business focused	**Data Architect**	**Data Analyst**

Role	Responsibility, framework actions and skills
Data Engineer	**Responsibility:** creation and upkeep of the **systems that store, process and move data**; many focus on the systems' **efficiency and extensibility**, ensuring capacity for **existing and exploratory or future workloads**. **Actions:** Ingest Data → Design & Refine Data → Optimize Data → Data Products & Services. **Skills:** background in **system administration**, plus a **fairly deep background in common data-processing algorithms and their implementation across various systems and tools**.
Data Architect	**Responsibility:** the **data in the refined and production stage locations** (occasionally raw too). **Objective: make this data accessible and broadly usable for a wide range of analyses.** Architects create **catalogs** to improve **discoverability and usability**, and create, apply and enforce **naming conventions and standard documentation practices**. **Actions:** Design & Refine Data → Optimize Data. **Skills:** a requirements-gathering pipeline — from **documented data needs** → an **abstract model** of where to source data and how to organise it → **concrete staging** via **data schemas and alignment conventions between schemas**. Needs fluency in the access and manipulation languages of a broad set of tools — variants of **SQL**, **Sqoop and Kafka**, analytics systems from **SAP and IBM** — plus standard conventions like **First, Second and Third Normal Forms**.
Data Scientist	**Responsibility: finding and verifying deep or complex sets of insights** — via **advanced statistical analyses** or **machine learning**, or by **conducting experiments such as modern A/B tests**; in some organisations also **"productionalizing"** those insights. **Actions:** Describe Data · Assess Data Utility · Design & Refine Data · Prototype Modeling · Optimize Data · Data Products & Services.

Role	Responsibility, framework actions and skills
	Two sub-types: *statistics-focused* (A/B testing; theory and practice of setting up and analysing experiments) and *engineering-focused* (prototyping and building data-driven services; software-engineering best practices for complex applications). **Shared skills:** the **mathematics and statistics algorithms** that reveal and test insights, plus the tools that apply them — **R, SAS, Python, SPSS**.
Analyst	**Responsibility: finding and delivering actionable insight**. Where data scientists take **exploratory, open-ended** analyses, analysts provide **critical information** — often **top-line metrics / KPIs** — delivered in **reports (regular and ad hoc)** or as **talking points to justify a decision**. **Actions:** Describe Data · Assess Data Utility · Design & Refine Data · Generate Ad-Hoc Reports · Optimize Data · Regular Reporting. **The blurry line with scientists:** analysts often go deeper — beyond identifying **correlated indicators of KPI trends**, performing **causal analysis** on those indicators. **Skills:** good mathematics and statistics plus **domain expertise**; more generically, analysts are **strong systems thinkers** — able to **connect insights that might be co-relevant and propose ways to measure the extent of their relationships**.

9.2 Roles across the nine actions

#	Action	Primary owner(s)
1	Ingesting data	**Data engineers** (focus on data systems)
2	Describing data	**Data engineers**
3	Assessing data utility (proprietary metadata)	**Analysts** (they hold the business knowledge); **data scientists** where data is particularly complex or messy
4	Designing & building refined data	**Data architects**; data engineers if storage/processing infrastructure needs modification and monitoring
5	Ad hoc reporting	**Analysts**
6	Exploratory modelling & forecasting	**Data scientists**
7	Designing & building optimised data	**Data architects + data engineers**
8	Regular reporting	**Analysts, with help from data engineers**
9	Building products & services	**Data scientists, with help from data engineers**

The caveat the book insists on: These associations are selective; in reality **people help wherever they can**. Engineers hold the deepest data-systems knowledge, architects the best cataloguing and design knowledge, analysts the most comprehensive domain understanding,

scientists the deepest statistics and ML background — **but many data projects can be sufficiently progressed with only cursory knowledge of some of these areas.**

9.3 The five organisational best practices

1. Provide **wide access to data**
2. Implement mechanisms to **track data usage**
3. Use a **common data manipulation language** spanning business units and user roles
4. Maintain a system allowing **easy transition from development to production**
5. Consider a **rotation program across roles** to enable cleaner hand-offs and increase cross-functional trust

(1) Wide access — "data democratization" / "self-service data analytics"

Not broad *overwrite* capability — but, within legal boundaries, **everyone in the organisation should be able to analyse the data you have**, because value creation is a funnel and you want as many people as possible finding insights. Two standard objections and the book's answers:

- *"The infrastructure is costly."* → In the authors' experience **the generated value more than compensates** for the additional cost.
- *"People will find conflicting insights, which slows us down."* → **(a)** Build robust refined datasets and drive the majority of analytics to source from them — this **mitigates superficial conflicts**. **(b)** Embrace the remaining conflicts and build practices addressing them directly: with superficial conflicts minimised, what remains largely represents **genuinely different views on how to measure or interpret the data**, and it **benefits the organisation to uncover those perspectives and coordinate them**. Robust insights **survive these interrogations**, and should be trusted **more** as a result.

(2) Track data usage

Improves your ability to **resolve conflicting insights**, and helps determine the **cost/benefit of altering refined datasets** (adding a dataset many people currently source from the raw stage; adding blended datasets used by many analyses).

The deeper reason — feedback loops: As organisations rely more on complex enrichments from inferences or predictions (regularly predicting churn and using it to drive operations), **tracking data movement becomes critical to enhancing or protecting the utility of those inferences.** Mechanism: the **inferred values shift the operations of the organisation, which shifts the data that gets collected** — the data is then shifting **partly in response to the inferences**. If the inference logic assumed the data had **no such feedback aspect**, the inferences become **inaccurate or biased over time**. *(Ref: Sculley et al., "Hidden Technical Debt in Machine Learning Systems," NIPS 2015.)*

(3–5) Common language, easy dev→production, rotation

- **(3) A common data manipulation language** is critical to **collaboration**: at minimum, people working together on one analysis must **share the basic tools**. Over time, people picking up **old analyses** benefit from being able to **immediately rerun the prior analysis and then modify it within the same language**. Today many organisations rely on **Excel or some flavour of SQL**.
- **(4) Easy dev → production transition.** Historically, exploration happened in one set of tools (**Excel, R, Python, SAS**) while production versions were rebuilt in **a more basic software-engineering framework**. More effective organisations have shifted to tools supporting **"productionalization" of exploratory logic more directly** — most newer tools simply **wrap exploratory scripts in a scheduling and monitoring framework**.
- **(5) Rotation program.** Superficially it **increases the breadth of skills** available. More fundamentally it **builds empathy and trust across job roles**, leading to **cleaner hand-offs on projects spanning multiple groups**.

10. Chapter 9 — Data Wrangling Tools

10.1 The three representatives

- **Excel** — a **spreadsheet application** letting users manipulate, analyse and store data in a **tabular** format.
- **SQL** — a **general-purpose tool for data manipulation**, and as such **widely embedded in many relational database distributions**.
- **Trifacta Wrangler** — a **visual data preparation application** for transforming **structured and unstructured** data and **profiling the output of those transformations** in an interactive interface.

Ages: Excel and SQL are both **more than 30 years old**, which the authors take as evidence of **the longevity of wrangling as a task**. **Trifacta Wrangler launched in 2012**, an outgrowth of academic research at **UC Berkeley and Stanford**. **Four comparison dimensions:** supported **data size**, required **infrastructure**, supported **data structures**, and **transformation paradigms**.

10.2 The comparison table

Dimension	Excel	SQL	Trifacta Wrangler
Max data size	MB to GB (files up to one or two gigabytes)	GB to TB (multi-terabyte production transaction datasets; **DB2 can scale to petabytes**)	**Unlimited** — megabytes to petabytes
Infrastructure	**Desktop** (personal computer)	**Server** (one or more networked servers)	**Cluster** — Hadoop cluster or a single server, with the **execution**

Dimension	Excel	SQL	Trifacta Wrangler
			environment determined automatically at runtime based on data volume and logical complexity
Data structures	**Grid** — need not be rectangular or completely filled	**Tabular (uniform records)** — must be rectangular and conform to a specific schema	**Various** — structured, semi-structured and unstructured
UI paradigm	Script **and wizards**	**Script only**	Script, **"builder,"** and **machine-guided transformation creation**
Transformation scope	**Single values** — formulas applying to single grid cells	**Programmatic** — a script applying to multiple records	**Programmatic** — a script applying to multiple records

Why size and infrastructure are linked: you wouldn't wrangle terabytes on a desktop application; conversely, using a **big-data distributed platform like a Hadoop cluster** for a few megabytes would be a **massively wasteful** use of computing power and budget.

10.3 Structures and paradigms in detail

Tool	Structure and transformation detail
Excel	Requires a **grid**, but the grid need **not be rectangular or completely filled**: people routinely put **multiple tables in one grid**, **mix descriptive text with data**, or **embed graphics**. Within each cell Excel supports a **wide variety of value types**. **Because of that heterogeneity, the single most important data element in Excel is the* cell*, not the record. The basic unit of transformation is a single grid cell, operated on by formulas; rows are sequential integers and columns are the Latin alphabet from "A" ("AA" follows "Z"). The common output of a formula is a single value in a single cell — the sum displays in the cell and the formula is embedded in the cell itself. Two mechanisms extend a transformation across cells: (1) updating references in formulas as you copy them; (2) array formulas, which additionally require the entire array of cells to be modified in a single pass, preserving consistency. Excel supports a complex series of calculations building on one another, but the steps are embedded throughout the data; standard practice is a new column per step, which keeps intermediate steps visible. If people retain only final values and delete the*

Tool	Structure and transformation detail
	*intermediate steps, then unless external documentation is created, the dataset preserves no lineage or history of how values were created.**
SQL	Expects **a set of records where every record contains the same set of fields**. Operates **at the level of entire records**; it is possible to isolate a specific field in a specific record, but more common uses **append or overwrite records**, or **select subsets for aggregations**. *Constraints it enforces:* **encoding restrictions on fields** (all values in a field share a datatype — an "integer" column cannot contain strings); and most systems require **designation of primary keys** and **enforce their unique occurrence**. *The verbosity problem: *any complete transformation statement in SQL requires specifying the fields involved, their output locations,* and *every other field that is "passing through" — in other words, every complete transformation in SQL requires specifying the entire schema of the dataset.* Two scripting styles: *a series of queries storing intermediate results — lets you interrogate intermediate values to assess validity, at the cost of managing those intermediate datasets; or a single long nested query — which makes it difficult to create and profile intermediate results, requiring valid variants of the query that output the intermediary instead of the final result**.
Trifacta Wrangler	Handles **structured, semi-structured and unstructured** data; data **need not be explicitly broken into rows and columns or fully populated**. Supports basic types (**integers, strings, Booleans**) and **more complex custom types such as dates, US states and phone numbers**. Operates at the level of **record fields* and *entire records. Transformations often look like subsets of SQL query statements — specifying just the fields involved and how they are combined; fields that pass through do not need to be specified, whereas in SQL they do, producing more readable scripts relative to SQL. Whole records can be operated on for filtering or aligning fields across records from different datasets during joins and unions.* Script properties: *all transformation logic sits in one place, so users don't click around cells to discover what logic was applied; scripts abbreviate passed-through fields; and users can click back through script steps to see the state of the dataset at intermediate points**.

10.4 Profiling support compared

Tool	Profiling support
Excel	Supports **reviewing individual cells** and **automated validation** of them. **Simple distributional checks are supported for numeric blocks** — sums, averages, minimums, maximums — **but not for strings or more complex**

Tool	Profiling support
	values like lists and arrays. Visual profiles (bar charts, scatterplots) **can be created but are time consuming.**
SQL	Some modern tools such as **Periscope and Looker** connect data visualisations **directly to SQL queries**. But **most SQL users profile by instantiating the dataset** — as an intermediary or final result — **and then applying simple SQL queries to view and summarise.**
Trifacta	**Profiling is embedded in the core user interface.** Two primary visual charts: one showing **validity satisfaction of individual field values**, the other showing **the distribution of values** — giving a sense of the **"shape" of the data**. The charts are **interactive**, letting the user **quickly select subsets of values and create transformations targeted at that subset.**

10.5 Choosing a tool

The book's answer: **There is no single correct choice.** It depends on **what you hope to do with your data.** The differentiators to weigh — *before* selecting a tool — are **data volume, infrastructure scale, and data structure**, plus the transformation and profiling paradigm that suits your team's common language.

11. Master Glossary

Term	One-line definition
Data wrangling	The process of transforming or preparing data for analysis; occurs *between* the data stages
ETL	One *type* of wrangling — the type managed and overseen by shared services / IT
Indirect value	Data influences people's decisions or inspires process change
Direct value	Data feeds automated systems that act without human input
Raw / Refined / Production	The three data stages: discover · make broadly usable · make narrowly optimal
Data value funnel	Many sources → many exploratory analyses → few valuable applications; conversion is never 100%
Schema-on-write	Data must satisfy structural/syntactic constraints to be ingested (warehouses)
Schema-on-read	Structure is not constructed or enforced until the data is used (HDFS/S3)
Generic metadata	Describes the data; broadly useful to anyone using the dataset
Custom/proprietary metadata	Contextualises generic metadata to a specific analysis or organisation
Structure	Format and encoding of records and fields (rectangular → jagged → heterogeneous)
Granularity	What one record represents; coarse vs fine
Accuracy	Quality/consistency of the values themselves
Temporality	A record represents an entity *at a time*; representations go stale
Scope	(a) attribute breadth; (b) population coverage and its bias
Imputation	Replacing inaccurate/missing values with default or estimated values
Profiling	Feedback that assures you a transformation succeeded; also how you learn what's in the data
Syntactic constraint	Constraint on the literal/format of a value
Semantic constraint	Constraint rooted in context or proprietary business logic
Set-based profiling	Profiling the shape/extent of a distribution, or relationships between fields

Term	One-line definition
Stratified sample	A sample covering every group/stratum; may bias overall trends by over-representing small groups
Structuring	Transformations that change form or schema
Enriching	Transformations that add fundamentally new values
Cleansing	Transformations that fix irregularities
Positional extraction	Extract a substring by fixed start/end positions
Complex positional extraction	Search for a delimiter to get the position, then slice (nested function)
Pattern extraction	Extract via a rule / regular expression
JSON array / JSON map	Ordered sequence in `[ ]` / key-value pairs in `{ }`
Column-to-row pivot	Unpivot / denormalize; wide → long
Row-to-column pivot	Field values become fields; long → wide; makes indicator variables
Union	Stack records vertically
Join	Link records horizontally on a key; inner / left outer / right outer / full outer
Fuzzy matching	Linking records despite misspellings or minor differences
UDF	User-Defined Function encoding proprietary derivation logic
Magic threshold	A concise, actionable KPI target that exposes the core dynamic of the system
Data democratization	Wide (read/analyse) access to organisational data; self-service analytics

11.1 Three transformation types vs two profiling types — one-glance grid

	Structuring	Enriching	Cleansing
Changes	Form / schema / granularity	Information content	Value correctness
Source of new values	Data already present	Data *not* previously present	Corrected / estimated versions of present values
Typical operations	Reorder, extract, combine, filter, aggregate, pivot	Union, join, insert metadata, derive	Drop nulls, impute, flag invalid, standardise

	Structuring	Enriching	Cleansing
Profiling that validates it	Set-based (did the grain/shape change as intended?)	Set-based (blend quality, link counts)	Individual-value (validity rates)