Open Access | https://doi.org/10.25080/tyyd7356

SciPy 2025

July 7 - July 13, 2025

Proceedings of the 24th Python in Science Conference
ISSN: 2575-9752

A Practical Guide to Data Quality: Problems, Detection, and Strategy

Edson Bomfim[1,2]

[1]Final04, [2]Universidade de São Paulo, USP

Abstract

The proliferation of data-driven systems, from machine learning models to business intelligence platforms, has placed unprecedented importance on the quality of the underlying data. The principle of "garbage in, garbage out" has never been more relevant, as poor data quality can lead to flawed models, incorrect business decisions, and a fundamental lack of trust in data systems.

This paper serves as a practical guide for software developers and data practitioners to understand, identify, and address data quality issues. We present a comprehensive taxonomy of 23 common data quality problems, synthesizing definitions and contexts from empirical software engineering and database literature. Subsequently, we catalogue 22 distinct methods for detecting these issues, categorizing them into actionable strategies: Rule-Based & Constraint Enforcement, Statistical & Machine Learning Analysis, Data Comparison & Standards, and Human & Manual Review.

By mapping specific problems to their detection-method categories, we provide a structured framework to help teams develop robust, efficient, and targeted data validation strategies.

We conclude by discussing how the systematic practice of "data testing," analogous to software testing, not only improves pipeline maintainability and dataset trustworthiness but also enhances the overall design and reliability of data-intensive systems.

Keywords data, data quality, software quality

1. Introduction

In the modern technological landscape, data is the foundational asset that powers innovation, from scientific research, training sophisticated machine learning (ML) models to driving critical business analytics. A dataset can be defined as a structured collection of individual data points, or instances, that hold information about a set of entities sharing some common characteristics. The utility and reliability of any system built upon this data are inextricably linked to its quality.

This paper treats the quality of data as the characteristic of the dataset to match the natural distribution, in other words the degree to which it accurately and faithfully represents the real-world phenomena it purports to describe [1], [2]. This is a crucial distinction. For example, a common challenge in machine learning is class imbalance, where one class is significantly underrepresented. However, if this imbalance accurately reflects the real world (e.g., fraudulent transactions are rare), the dataset itself is not of poor quality; rather, it is a perfect representation of a difficult problem. The challenge then lies with the modeling technique, not the data's fidelity. Our focus is on errors where the data *fails* to match the natural distribution.

Published Jul 10, 2025

Correspondence to
Edson Bomfim
research@final04.com

The consequences of poor data quality are severe, leading to the well-known "garbage in, garbage out" paradigm. Flawed data can introduce subtle biases in algorithms, generate misleading analytical reports, and erode stakeholder trust, ultimately undermining the value of data-driven initiatives [3], [4]. Despite its importance, a systematic, developer-focused approach to identifying and mitigating these issues is often lacking.

This paper aims to fill that gap by providing a practical, empirical guide for data practitioners. We synthesize a comprehensive taxonomy of data quality problems and a corresponding catalogue of detection methods, drawing from extensive research in empirical software engineering, data cleaning, database management and the author own experience. The goal is to equip developers with a structured framework to reason about, test for, and strategically address data quality within their systems.

The paper is structured as follows:

- Data Quality Problems details a taxonomy of 23 common data quality problems, providing concise definitions and relevant context for each, and organizes them into practical categories.
- Methods to Detect Data Quality Problems presents a catalogue of 22 methods used to detect these problems, discusses their automation potential, and groups them into four strategic categories. This section includes a mapping of which method categories are effective for which errors.
- Case Study: A Practical Analysis of a Public Health Dataset provides a practical analysis of a real-world public health dataset, demonstrating how the catalogued problems and detection methods can be applied to identify concrete data quality issues.
- Conclusions discuss with the strategic implications of adopting a proactive data quality assurance mindset, drawing parallels between data testing and established software testing practices to argue for a more robust approach to building data systems.

2. Data Quality Problems

A data quality problem, or "dirty data," is any instance where the data fails to correctly represent the real-world entity or event it describes [5], [6], [7]. These problems can be subtle or overt, but all carry the risk of corrupting analysis and undermining model performance. Below is a taxonomy of common data quality problems compiled from the literature.

2.1. Taxonomy of Data Quality Problems

1. **Missing Data / Incompleteness / Empty Values:** Records or fields that should contain a value but are null, empty, or otherwise not populated. This is one of the most common issues, forcing decisions about imputation or removal [8], [9].
2. **Duplicates / Redundancy:** The same real-world entity is represented by two or more records in a dataset. These can be exact duplicates or approximate (fuzzy) duplicates with minor variations [7], [10], [11].
3. **Inconsistency / Contradicting Records / Semantic Consistency Violations:** A single real-world entity is described by multiple records that contain conflicting information (e.g., the same customer with two different birth dates). This also includes violations of semantic rules (e.g., a patient's discharge date is before their admission date) [7], [12].
4. **Wrong / Incorrect Data / Invalid Data:** Data values that are syntactically valid but factually wrong (e.g., an age of "30" for a person who is actually 40). This is one of the hardest problems to detect without an external source of truth [6], [10].

5. **Misspellings:** Typographical errors in textual data, which can hinder matching and categorization (e.g., “Jhon” instead of “John”) [6], [10].
6. **Outdated Temporal Data / Timeliness / Currency / Volatility:** The data was correct at the time of recording but is no longer valid due to real-world changes (e.g., a customer’s address before they moved). This dimension measures how up-to-date the data is [6], [13].
7. **Non-standardized / Non-Conforming Data / Irregularities:** The use of different formats or units to represent the same information (e.g., “USA”, “U.S.A.”, “United States”; or measurements in both Celsius and Fahrenheit) [6], [6].
8. **Ambiguous Data / Incomplete Context:** Data that can be interpreted in multiple ways without additional context. This can arise from abbreviations (“J. Stevens” for John or Jack) or homonyms (“Miami” in Florida vs. Ohio) [6], [10].
9. **Embedded / Extraneous Data / Value Items Beyond Attribute Context:** A single field contains multiple pieces of information that should be separate (e.g., a “name” field containing “President John Stevens”) or information that belongs in another field (e.g., a zip code in an address line) [6], [10].
10. **Misfielded Values:** Correct data is stored in the wrong field (e.g., the value “Portugal” in a “city” attribute) [10].
11. **Outliers / Noise:** Erroneous data points that lie significantly outside the typical distribution of values, often due to measurement or entry errors. They are distinct from valid but extreme data points [8].
12. **Insufficient / Imprecise Metadata:** The metadata (data about the data) is missing or does not adequately describe the metrics, entities, or collection process, leading to a high risk of misinterpretation [13], [14].
13. **Schema Violations / Wrong Data Type / Structural Conflicts:** Data that violates the formal schema of the database, such as incorrect data types (e.g., “XY” in an age field), or different structural representations for the same object across systems [7], [10].
14. **Dangling Data / Referential Integrity Violation:** A record refers to another record that does not exist, breaking a relational link (e.g., an order record with a `customer_id` that does not exist in the customers table) [6], [10].
15. **Concurrency Control / Transaction Issues:** Data corruption arising from the lack of proper transaction management in databases, leading to issues like lost updates or dirty reads. While typically handled by the DBMS, it is a source of poor data quality [6].
16. **Wrong Categorical Data:** A value for a categorical attribute is outside the predefined set of valid categories (e.g., `shipping_method` is “air” when the only valid options are “ground” or “sea”) [6].
17. **Name Conflicts / Homonyms / Synonyms:** When integrating data, the same name is used for different objects (homonyms) or different names are used for the same object (synonyms), causing structural ambiguity [7].
18. **Different Aggregation Levels:** In data integration, the same entity is represented at different levels of detail across sources (e.g., sales data aggregated by day in one source and by month in another) [15].
19. **Circularity Among Tuples in Self-Relationship:** A logical impossibility where entities in a self-referencing relationship form a cycle (e.g., Product A is a sub-product of B, and Product B is a sub-product of A) [15].
20. **Semi-empty Tuple:** A record where a large number of its attributes are empty, but not all. This may indicate a partial data entry failure or a distinct sub-type of entity that was not modeled correctly [15].
21. **Data Miscoding:** The incorrect assignment of data types during processing, such as treating a numerical feature as a string. This is a process-induced error rather than a source data error [16].

22. **Distribution Shift / Data Drift:** The statistical properties of the data change over time or between different contexts (e.g., training vs. production), rendering models trained on older data less accurate [16].
23. **Plausibility:** Data values that are syntactically correct and within a valid range, but are highly improbable in the given context [17].

2.1.1. *A Note on Excluded Concepts*

- **Trust/Trustworthiness:** While critically important, trust is a property of the relationship between a data consumer and a data source, not an intrinsic property of the dataset itself. It is often established over time through repeated positive interactions and is influenced by the provenance and perceived reliability of the source [8]. Therefore, it is considered outside the scope of direct data quality measurement.
- **Accessibility/Availability/Security:** These dimensions relate to the systems and procedures governing access to the data, not the content of the data itself [13]. While a dataset that cannot be accessed is useless, we focus here on the quality of the data once it has been accessed.
- **Representational Validity/Sampling Bias:** A dataset can be internally accurate yet fail to represent its target phenomenon. This issue, known as sampling bias, stems from fundamental flaws in the data collection strategy—for instance, surveying only a specific subset of a population. Because these are problems of study design concerning data that was never collected, they fall outside this paper's scope, which is confined to the intrinsic quality and fidelity of the data at hand.

2.2. Detection Scalability

Data quality problems can be broadly divided into two groups based on the scope required for their detection:

- **Single-Datapoint Errors:** These errors can be identified by examining a single record or data point in isolation. Examples include `Missing Data`, `Schema Violations`, and `Misspellings`. Detection for these errors is highly parallelizable and scales linearly with the size of the dataset, as each data point can be checked independently.
- **Multi-Datapoint Errors:** These errors require the context of other records, or even the entire dataset, to be identified. `Duplicates`, `Outliers`, and `Distribution Shift` fall into this category. Detection can be computationally expensive, often requiring full-dataset statistics. While optimizations like sorting or indexing can help, adding new data may necessitate a full re-evaluation of the entire dataset.

2.3. Categorizing Data Quality Problems

While a flat list of problems is useful, categorization helps in forming a mental model for tackling them. A common approach in the literature is to group them by the aspect of quality they violate [13]:

- **Content Quality (Accuracy & Integrity):** Problems related to the correctness and internal consistency of the data. Includes *Inconsistency*, *Wrong Data*, *Misspellings*, *Outliers*, *Schema Violations*, and *Dangling Data*.
- **Completeness (Sufficiency of Information):** Problems concerning missing information. Includes *Missing Data*, *Semi-empty Tuple*, and *Insufficient Metadata*.
- **Representational & Structural Quality (Form & Understandability):** Problems with how the data is structured and represented. Includes *Non-standardized Data*, *Embedded Data*, *Misfielded Values*, *Name Conflicts*, and *Wrong Categorical Data*.

- **Timeliness & Relevance (Fitness for Current Context):** Problems related to the data being out-of-date or contextually inappropriate. Includes *Outdated Temporal Data* and *Distribution Shift*.

While this categorization is useful for understanding the *impact* of poor data quality, a more practical approach is to categorize problems based on the *methods required to detect them*. This pragmatic view directly informs the implementation of a data quality assurance strategy, which we explore in the next section.

3. Methods to Detect Data Quality Problems

Detecting the problems outlined in Section 2 requires a diverse toolkit of methods, ranging from simple, deterministic rules to complex statistical models. Below, we catalogue these methods and discuss some aspects of their application.

Table 1. *Mapping Detection Methods to Data Quality Problems*

Method	Applicable Data Quality Problems Found
Database Constraints (Not Null, Unique, Primary Key, Foreign Key)	Missing Data, Duplicates, Dangling Data, Schema Violations
User-Specified Integrity Constraints & Triggers	Inconsistency, Wrong Data, Wrong Categorical Data, Circularity
Direct Checks (for Null, Empty Strings, Dummy Values)	Missing Data, Semi-empty Tuple
Quantitative & Statistical Metrics	Missing Data (e.g., completion rate), Distribution Shift
Comparison & Similarity (Direct, Distance, String Metrics)	Duplicates, Inconsistency, Misspellings, Non-standardized Data
Sorted Neighbourhood Method	Duplicates (approximate)
Attribute Cardinality Checks	Duplicates (where cardinality should equal row count)
Redundancy Analysis	Redundancy (correlated features)
Semantic & Rule-Based Testing (incl. Regex, Set Validation)	Inconsistency, Wrong Data, Misfielded Values, Schema Violations, Non-standardized Data
Probability-based Metrics & Statistical Tests	Inconsistency (Semantic), Plausibility
Association Rule Mining	Inconsistency, Wrong Data
Comparison to Reference Data (Real World, Legal Values)	Wrong / Incorrect Data, Outdated Temporal Data
Explicit Type Checking & Domain Format Checks	Schema Violations, Non-standardized Data
Spell Checker	Misspellings
Temporal Constraint Checks	Outdated Temporal Data
Distribution Analysis (L-infinity, Jensen-Shannon, etc.)	Distribution Shift, Outliers
Outlier Detection Algorithms (ML, Statistical, Distance-based)	Outliers, Noise
Clustering Algorithms	Outliers, Duplicates (by grouping similar records)
Lookup Tables & Dictionaries (Thesaurus, Name/Address Dirs)	Non-standardized Data, Ambiguous Data, Misspellings

Metadata & Schema Evaluation (Metamodels, DQRs)	Insufficient / Imprecise Metadata, Schema Violations, Name Conflicts
Cycle Detection Algorithms	Circularity Among Tuples
Data Profiling & Human Review	All (serves as a general-purpose discovery and verification method)

3.1. Automation and Maintenance:

The methods above vary significantly in their potential for automation. Database constraints are fully automated, while methods like statistical analysis and rule-based testing can be incorporated into automated data pipelines. Others, such as resolving ambiguous data using a thesaurus or verifying incorrect data against the real world, fundamentally require human-in-the-loop intervention.

Just as software testing is an integral part of CI/CD pipelines, data quality tests should be an integral part of data pipelines. Whenever data is updated, ingested, or transformed, a corresponding suite of data tests should be executed. Single-datapoint checks can be run efficiently on new or changed data. However, for multi-datapoint checks, an incremental update may not be sufficient; a full-dataset re-evaluation is often necessary to ensure issues like new duplicates or shifts in the overall distribution are caught.

3.2. Categories of Detection Methods

To provide a clear, actionable framework, we first present a table mapping individual detection methods to the data quality problems they are primarily used to find. A more strategic approach is to group these methods into broader categories. We propose four categories of data quality detection methods, which group the 22 techniques into strategic approaches:

3.2.1. *Rule-Based & Constraint Enforcement*

This category involves defining explicit, deterministic rules that data must adhere to, often enforced directly at the database level through mechanisms like schema constraints. These methods are highly automatable and effective for catching violations of known business logic, structural integrity, and formatting requirements, such as ensuring a field is never null, conforms to a specific data type, or follows a predefined pattern.

3.2.2. *Statistical & Machine Learning Analysis*

This approach leverages mathematical and algorithmic techniques to identify data quality issues by analyzing the aggregate properties and distributions within a dataset. By establishing statistical norms, these methods can automatically flag outliers and uncover improbable or inconsistent data patterns that would not violate simple, hard-coded rules.

3.2.3. *Data Comparison & Standards*

This category focuses on validating data by comparing it against a trusted source of truth or by identifying inconsistencies through intra-dataset comparisons. It includes techniques like matching records against external reference data (e.g., a list of valid postal codes), using string similarity metrics to find fuzzy duplicates, and employing lookup tables or dictionaries to standardize terminology.

3.2.4. *Human & Manual Review*

This category encompasses all methods that rely on human intelligence, domain expertise, and contextual understanding to identify and verify data quality problems. It serves as the final line of defense for detecting complex, subtle, or novel issues that automated systems miss, such as evaluating the sufficiency of metadata, verifying the real-world accuracy of a questionable value, or interpreting ambiguous data through expert judgment.

Table 2. *Categories of Data Quality Detection Methods*

Category	Methods Included
Rule-Based & Constraint Enforcement	Not Null Constraints, Unique/Primary Key Constraints, General DB Integrity Constraints, Triggers, Rule-based Testing, Regex Pattern Verification, Set Validation, Type Checking/ Constraints, Temporal Constraints, Domain Format Checks, Cycle Detection Algorithms, Formal Specifications (DQRs, DPRs).
Statistical & Machine Learning Analysis	Quantitative Metrics, Statistical Criterion/Sampling, Redundancy Analysis, Probability-based Metrics, Association Rule Mining, Distribution Analysis (L-infinity, Jensen-Shannon, etc.), All Outlier Detection Methods (ML, Statistical, Distance-based), Clustering, Trimmed Means.
Data Comparison & Standards	Direct Comparison, Distance Metrics, String Similarity Metrics, Sorted Neighbourhood, Comparison to Reference Data, Standardization/ Normalization, Lookup Tables/Dictionaries, Thesaurus, Name/Address Directories.
Human & Manual Review	Identifying Dummy Values, Data Profiling, Metadata Evaluation, Human Verification/Manual Inspection, Domain Expert Intervention.

This categorization allows for a structured approach to building a data quality firewall. A team can start with the most automatable category, “Rule-Based & Constraint Enforcement,” and progressively add more sophisticated “Statistical & ML Analysis” and “Data Comparison” methods, while reserving “Human & Manual Review” for exceptions and complex cases.

Also we present the Table 3 which provides a high-level map for practitioners, showing which categories of methods are effective at identifying specific data quality problems.

Table 3. *Linking Data Problems to Detection Method Categories*

Data Quality Problem	Rule-Based & Constraint	Statistical & ML	Data Comparison & Standards	Human & Manual
Missing Data / Incompleteness	✓	✓		✓
Duplicates / Redundancy	✓	✓	✓	✓
Inconsistency / Contradictions	✓	✓		✓
Wrong / Incorrect Data	✓	✓	✓	✓

Misspellings				✓	✓
Outdated Temporal Data	✓			✓	✓
Non-standardized Data	✓			✓	✓
Ambiguous Data				✓	✓
Embedded / Extraneous Data	✓				✓
Misfielded Values	✓				✓
Outliers / Noise			✓		✓
Insufficient / Imprecise Metadata					✓
Schema Violations	✓				✓
Dangling Data / Referential Integrity	✓				
Concurrency / Transaction Issues	✓				
Wrong Categorical Data	✓		✓		✓
Name Conflicts					✓
Different Aggregation Levels				✓	✓
Circularity Among Tuples	✓				
Semi-empty Tuple	✓				✓
Data Miscoding	✓				✓
Distribution Shift / Data Drift			✓		✓
Plausibility			✓		✓

4. Case Study: A Practical Analysis of a Public Health Dataset

To bridge the gap between the theoretical taxonomy of problems and the catalogue of detection methods, this section presents a practical analysis of a real-world dataset: Brazil's Mortality Information System (SIM - Sistema de Informação sobre Mortalidade) [18]. As a large, publicly available dataset aggregated from numerous sources across the country, it serves as an excellent example of the types of data quality challenges practitioners face. We will examine data from 2024 to illustrate how specific problems can be identified using the methods discussed. This dataset is publicly available from Brazil's Open Data SUS portal.

Below is a sample from the SIM dataset, showing a selection of key fields.

Table 4. *Sample Data from the Brazilian Mortality Information System (SIM) 2024*

DTOBITO (Date of Death)	HORAOBITO (Time of Death)	DTNASC (Date of Birth)	IDADE (Age)	SEXO (Sex)	RACACOR (Race/ Color)	ESTCIV (Marital Status)	SERIESCFAL (Last Grade)	CAUSABAS (Basic Cause)	ATESTADO (Death Cert. Causes)
19042024	2301	24121964	459	1	1	1		I279	J81/ I500/ I279/ I10
18012024	1030	29041958	465	2	1	2		A09	I219/ A09
22012024	0700	23031939	484	2	4	2		E112	I219/ I10/ E112
12012024	0800	08031950	473	2	2	5		I219	I219/ I509/ I10
01032024	1030	12051938	485	2	1	9		N19	J969/ I469/ D649/ N19*I509

4.1. Example 1: Detecting Data Miscoding with Human & Manual Review

Problem: A preliminary inspection of the IDADE (Age) field reveals values such as 459 and 484. Without domain context, these appear to be erroneous. A simple cross-validation by subtracting DTNASC from DTOBITO for the first record reveals an age of 59 years, suggesting the value 459 is not a raw integer but an encoded value.

Detection: This is a classic case where Human & Manual Review, informed by domain knowledge, is the primary detection method. The initial observation of impossible age values prompts a deeper investigation. Consulting the official data dictionary confirms that the IDADE field uses a special encoding where the leading digit signifies the time unit (e.g., ‘4’ for years) and the subsequent digits represent the quantity. Failure to identify this results in Data Miscoding, a severe process-induced error. Once detected, the remediation is a straightforward Rule-Based transformation.

4.2. Example 2: Quantifying Missing Data with Rule-Based Enforcement

Problem: The SERIESCFAL (Last Grade) column is blank in every record of the sample, representing a clear case of Missing Data. The impact of this problem depends entirely on the intended use case.

Detection: A simple Rule-Based check can identify null or empty values on a per-record basis. However, the true utility comes from applying this rule across the entire dataset to establish a data quality metric. For an analysis where education level is a critical feature, a team might define an acceptance criterion, such as requiring at least 50% of records to be non-null. The following programmatic check implements this quality gate:

```
# Assumes 'df' is a pandas DataFrame of the full 2024 dataset
acceptance_threshold = 0.5
completeness_ratio = df.SERIESCFAL.notnull().sum() / len(df)
assert completeness_ratio >= acceptance_threshold
```

Executing this check against the full dataset would return False, indicating that the data fails this specific quality test and is not fit for this particular use.

4.3. Example 3: Identifying Inconsistency with Rule-Based Validation

Problem: A dataset can contain values that are individually valid but logically impossible in combination. According to the SIM documentation, SEXO=1 indicates a person born male, while GRAVIDEZ=1 signifies the person has been pregnant.

Detection: This Inconsistency can be detected using Rule-Based & Constraint Enforcement that evaluates a multi-field invariant. The following test efficiently isolates all records that violate this semantic rule:

```
# Filters for records where a person born male is recorded as having been pregnant
inconsistent_records = df[(df.GRAVIDEZ == 1) & (df.SEXO == 1)]
```

This test effectively detects that an error exists in the record but cannot determine which of the two fields is incorrect. The remediation strategy—whether to nullify the fields or discard the record—depends on further analysis or predefined business rules.

4.4. Example 4: Assessing Plausibility using Statistical Analysis

Problem: Certain data points, while not strictly impossible, may be so statistically improbable that they warrant scrutiny. The IDADEMAE (Mother's Age) field provides a clear example. Detection: By employing Statistical Analysis, we can profile the distribution of IDADEMAE to identify anomalous values. The table below shows the age distribution in bins for all non-null entries in the 2024 dataset.

Table 5. *Distribution of Mother's Age (IDADEMAE) in SIM 2024*

Age Bin	Count
(0, 10]	4
(10, 20]	4867
(20, 30]	11800
(30, 40]	7705
(40, 50]	1240
(50, 60]	10
(60, 70]	0
(70, 80]	0
(80, 90]	0
(90, 100]	26

This distribution immediately highlights potential Plausibility issues at the tails. While pregnancies in the 10-20 age group are common, the 4 instances below age 10 are highly suspect.

Even more striking are the 26 recorded pregnancies for mothers aged over 90. While not theoretically impossible with modern medicine, these are statistically extreme outliers and

highly likely to be data entry errors. Such records should be flagged as low-quality and considered for exclusion from sensitive analyses.

4.5. The Role of Data Quality Frameworks

Except for the Human & Manual check above, all others programmatic checks can be systemized using dedicated data quality frameworks. Great Expectations [19] is a open-source tool that institutionalizes "data testing" by allowing teams to declare assertions about their data in a readable, JSON-based format called "Expectations." This approach formalizes the data quality rules, making them version-controllable, shareable, and executable within automated pipelines, directly addressing the need for systematic data testing discussed in this paper.

The library provides a rich, built-in vocabulary of Expectations that directly map to the detection methods and problems we have catalogued. For instance, `expect_column_values_to_not_be_null` addresses Missing Data, `expect_column_values_to_be_in_set` handles Wrong Categorical Data, and `expect_column_mean_to_be_between` can detect Distribution Shift. More complex, multi-field invariants, such as the SEXO/GRAVIDEZ inconsistency, can be implemented using custom Expectations.

A key advantage of Great Expectations is its ability to automatically generate "Data Docs" human-readable documentation that presents the results of data validation runs. This feature transforms abstract quality metrics into tangible reports, fostering trust and communication between data producers and consumers. By integrating such a framework, the analysts can operationalize the principles outlined here, moving from ad-hoc data cleaning to a proactive, scalable, and automated data quality assurance strategy.

5. Conclusions

This paper has provided a comprehensive, practical-focused framework for understanding, identifying, and addressing data quality problems. By cataloguing 23 distinct types of "dirty data" and 22 corresponding detection methods, we have created a guide that can the turned into actionable strategies.

The key takeaway is the need to treat data quality assurance as a first-class citizen in the development of data systems. Understanding the fundamentals—the specific ways data can be dirty and the array of tools available to detect it—allows a team to optimize its validation strategy. Instead of applying a generic, one-size-fits-all approach, they can select the most efficient methods for their specific risks, whether that is enforcing a strict schema with **Rule-Based** methods, discovering novel anomalies with **Statistical** analysis, or standardizing inputs with **Data Comparison** techniques.

Integrating these methods into automated "data testing" suites within data pipelines is critical for long-term maintainability and trust. Just as unit tests verify code logic, data tests verify data integrity. This practice provides continuous feedback, catches regressions when data sources or processing logic change, and builds confidence in the final datasets used for researches, analytics, and machine learning.

Perhaps the most profound benefit of this approach lies in the design phase. The very process of defining data quality requirements and designing data tests forces a team to deeply consider potential failure modes. It requires them to ask critical questions: What are the invariants of our data? What are the semantic rules that must hold? What are the statistical norms? This process, much like writing the design of a prospective study or writing a formal specification for software, clarifies assumptions and exposes ambiguities *before* a

single line of data processing code is written. The resulting data system is not only more robust but also better designed, because it is built on a foundation of clearly articulated expectations about the data it will handle.

References

[1] Y. Wand and R. Y. Wang, "Anchoring data quality dimensions in ontological foundations," *Commun. ACM*, vol. 39, no. 11, pp. 86–95, 1996, doi: 10.1145/240455.240479.

[2] N. Hynes, D. Sculley, and M. Terry, "The Data Linter: Lightweight Automated Sanity Checking for ML Data Sets," 2017. [Online]. Available: http://learningsys.org/nips17/assets/papers/paper_19.pdf

[3] B. Saha and D. Srivastava, "Data quality: The other face of Big Data," in *2014 IEEE 30th International Conference on Data Engineering*, 2014, pp. 1294–1297. doi: 10.1109/ICDE.2014.6816764.

[4] K. Kerr, T. Norris, and R. Stockdale, "Data quality information and decision making: A healthcare case study," *ACIS 2007 Proceedings - 18th Australasian Conference on Information Systems*, 2007.

[5] W. Kim, B.-J. Choi, E. Hong, S.-K. Kim, and D. Lee, "A Taxonomy of Dirty Data," *Data Min. Knowl. Discov.*, vol. 7, pp. 81–99, 2003, doi: 10.1023/A:1021564703268.

[6] H. Müller and J.-C. Freytag, "Problems, methods, and challenges in comprehensive data cleansing," 2003.

[7] J. Barateiro and H. Galhardas, "A Survey of Data Quality Tools.," *Datenbank-Spektrum*, vol. 14, pp. 15–21, 2005.

[8] M. Bosu and S. MacDonell, "A Taxonomy of Data Quality Challenges in Empirical Software Engineering," in *Proceedings of the Australian Software Engineering Conference, ASWEC*, 2013, pp. 97–106. doi: 10.1109/ASWEC.2013.21.

[9] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data Quality: Some Comments on the NASA Software Defect Datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013, doi: 10.1109/TSE.2013.11.

[10] H. Galhardas, D. Florescu, D. Shasha, and E. Simon, "AJAX: an extensible data cleaning tool," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, in SIGMOD '00. Dallas, Texas, USA, 2000, p. 590. doi: 10.1145/342009.336568.

[11] A. E. Monge and C. P. Elkan, "The field matching problem: Algorithms and applications," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, in KDD'96. Portland, Oregon, 1996, pp. 267–270.

[12] B. Heinrich, M. Klier, A. Schiller, and G. Wagner, "Assessing data quality – A probability-based metric for semantic consistency," *Decision Support Systems*, vol. 110, pp. 95–106, 2018, doi: https://doi.org/10.1016/j.dss.2018.03.011.

[13] F. Ridzuan and W. M. N. W. Zainon, "A Review on Data Quality Dimensions for Big Data," *Procedia Computer Science*, vol. 234, pp. 341–348, 2024, doi: https://doi.org/10.1016/j.procs.2024.03.008.

[14] Y. E. Ouatiti, M. Sayagh, N. Kerzazi, B. Adams, and A. E. Hassan, "The impact of concept drift and data leakage on log level prediction models," *Empirical Software Engineering*, vol. 29, no. 5, 2024, doi: 10.1007/s10664-024-10518-9.

[15] P. Oliveira, F. Rodrigues, and P. Rangel Henriques, "A Formal Definition of Data Quality Problems.," in *Proceedings of the 2005 International Conference on Information Quality, ICIQ 2005*, 2005.

[16] A. Bhatia, D. Lin, G. K. Rajbahadur, B. Adams, and A. E. Hassan, "Data Quality Antipatterns for Software Analytics." [Online]. Available: https://arxiv.org/abs/2408.12560

[17] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *Proceedings of the 2003 International Conference on Information Integration on the Web*, in IIWEB'03. Acapulco, Mexico, 2003, pp. 73–78.

[18] Brasil, Ministério da Saúde, "Banco de dados do Sistema´Unico de Saúde-DATASUS." [Online]. Available: https://opendatasus.saude.gov.br/pt_BR/dataset/sim

[19] A. Gong, J. Campbell, and Great Expectations, "Great Expectations." [Online]. Available: https://github.com/great-expectations/great_expectations