Automated Data Quality Validation for Smart City Data Ecosystems

, Moamin B. Abughazala†, Henry Muccini†,
†*DISIM Department*, *University of L'Aquila*, L'Aquila, Italy

***Abstract*—Smart cities generate massive volumes of heterogeneous data from sources such as traffic systems, environmental sensors, public transport, and citizen applications. Ensuring the quality of this urban data is crucial for reliable analytics, service optimization, and policy-making. However, data validation in smart city systems remains largely manual, error-prone, and non-scalable due to frequent schema evolution and variable data standards across departments.**

In this paper, we utilize the DQGen framework for automating data quality validation in smart city environments. Leveraging metadata extracted from open urban datasets, the framework maps standard quality dimensions—such as completeness, consistency, validity, and timeliness—to executable validation rules using Great Expectations. The generated scripts can be integrated into city dashboards or batch pipelines, allowing for continuous, transparent, and repeatable validation across evolving datasets.

We validate the framework using datasets from a municipal open data portal, which include traffic flow, air quality, and public transportation usage records.

***Index Terms*—Data Architecture, Smart Cities, Data-Quality**

I. Introduction

Smart cities rely on diverse and dynamic datasets generated by traffic sensors, environmental monitors, mobility platforms, and citizen applications. These datasets enable a wide range of services, including real-time traffic management, pollution monitoring, and urban planning. However, the increasing heterogeneity, volume, and velocity of urban data pose serious challenges to ensuring data quality [1]. Inconsistent formats, missing values, schema drift, and poorly validated fields can lead to inaccurate analytics, suboptimal decisions, and reduced citizen trust [2] [3].

Traditional data validation in smart city systems is often manual and fragmented across departments or vendors [4], leading to inefficiencies and non-reproducible quality assurance processes. To address this, we adopt DQGen, a metadata-driven framework designed initially for data-intensive applications, and adapt it for urban data environments. The DQGen system automates, scales, and standardizes data quality validation for smart city datasets by utilizing metadata and mapping standard quality dimensions to actionable rules.

II. Background

Urban data is typically structured in relational or semi-structured formats and is updated at varying temporal frequencies. Ensuring its quality is critical for operational systems (e.g., traffic control), predictive analytics (e.g., pollution forecasts), and citizen-facing dashboards.

While rule-based validation tools such as Great Expectations (GE) [5] offer a powerful foundation for data profiling and testing, they require manual configuration, limiting their applicability across evolving schemas and large datasets. Frameworks like DQGen automate this process by:

1) Extracting metadata (e.g., schema structure, data types),
2) Mapping quality dimensions (e.g., completeness, consistency) to validation rules, and
3) Generating executable validation code in Python using GE.

The original DQGen framework showed high effectiveness in industrial telecom datasets. This work explores its applicability in smart city data contexts, where schema evolution and multi-source integration are prevalent.

III. Methodology

A. DQGen Framework

DQGen framework automates data quality validation for smart city datasets from open data portals, which are typically tabular and cover domains like mobility, environment, and public services.

B. Metadata Extraction

The process begins with the automatic parsing of dataset metadata. DQGen supports structured input formats (e.g., CSV files or relational database exports) and extracts schema information such as:

- Table and column names
- Data types (integer, float, string, datetime, etc.)
- Nullability and primary key constraints (if available)

This metadata forms the basis for defining applicable validation rules in a scalable and schema-aware manner. For example, fields inferred as identifiers or time stamps are automatically assigned specific quality dimensions.

C. Dimension Mapping

Once metadata is extracted, the framework maps dataset columns to a set of configurable data quality dimensions, each targeting common integrity issues in urban data systems. These dimensions ensure robust validation across diverse domains:

1) Completeness verifies that mandatory fields (e.g., timestamp, station_id) are populated, which is essential for accurate analysis and downstream processing.

moamin.abughazala@graduate.univaq.it
henry.muccini@univaq.it

2) Uniqueness ensures that key identifiers, such as sensor_id or record_id, do not contain duplicates, preventing over-counting and ensuring referential integrity.
3) Validity checks whether values conform to expected data types or domains, such as enforcing that datetime fields are properly formatted or numeric fields do not contain text.
4) Timeliness assesses whether temporal values fall within plausible or predefined time windows, helping detect stale data or misaligned logging intervals.
5) Consistency enforces logical coherence across fields, such as ensuring arrival_time occurs after departure_time, or that calculated durations match timestamp differences.

This mapping process allows DQGen to generate targeted and reusable validation rules tailored to the semantics of urban datasets.

This mapping process is rule-driven but requires no manual scripting. Rules are chosen based on column semantics inferred from naming conventions or external profiles.

D. Code Generation

After mapping, the framework generates Python validation scripts using the Great Expectations (GE) library. Each script is customized based on:

- The table and column structure
- The mapped quality dimensions
- Templated GE expectations

For example, a column mapped to "completeness" will be checked using GE's expect_column_values_to_not_b_null function, while a uniqueness constraint triggers expect_column_values_to_be_unique. These expectations are filled into pre-defined script templates with placeholders for schema-specific details.

The final scripts can be scheduled as batch jobs or integrated into continuous data ingestion pipelines. Outputs include:

- Human-readable validation summaries
- JSON-based validation reports
- Alerts for failed expectations

E. Case Study

We selected three datasets from a municipal open data portal [6]:

- Traffic Flow (vehicle counts per hour and direction)
- Air Quality (PM2.5, NO2, CO2 measurements)
- Public Transport Usage (bus stop entries/exits)

Each dataset underwent metadata extraction, followed by the generation and execution of validation rules via DQGen-Smart. The output included readable validation reports and alerts for failed expectations.

IV. Results

To evaluate the applicability of DQGen in the smart city context, we conducted a pilot study using publicly available datasets from a municipal open data portal. Three datasets were selected from distinct urban service domains:

- Traffic Flow Dataset: Hourly vehicle counts from road sensors, segmented by direction and location.
- Air Quality Dataset: Periodic pollutant readings (PM2.5, CO2, NO2) from fixed monitoring stations.
- Public Transport Usage Dataset: Daily passenger counts at bus and tram stops, recorded by fare collection systems.

Each dataset was subjected to the full DQGen pipeline—from metadata extraction to script generation and rule execution.

TABLE I
Comparison of Manual Approach and DQGen for Smart City Datasets

Metric	Manual Approach	DQGen
Average validation script setup time	1–2 days	¡ 15 minutes
Required fields with completeness checks	˜65% coverage	100% coverage
Schema adaptability	Manual rework	Automatic
Consistency of rule logic	Medium	High

We evaluated DQGen against manual validation approaches typically used by data engineering teams. As summarized in Table I, the framework consistently generated executable GE scripts in under 15 minutes across all datasets. These scripts produced detailed validation reports in both HTML and JSON formats, highlighting failure rates and providing descriptive summaries for each expectation.

The generated scripts were deployed in a daily batch workflow, producing validation reports and automated alerts for failed checks, which enabled prompt issue resolution by both engineers and planners.

V. Conclusion

This work adapts the DQGen framework to smart city datasets, automating validation script generation based on metadata and quality dimensions. It reduces manual effort and improves consistency across diverse urban data. Real-world results demonstrate its scalability and transparency. Future directions include support for semi-structured data and domain-specific validation.

References

[1] Leo L Pipino, Yang W Lee, and Richard Y Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[2] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. Engineering ai systems: A research agenda. *Artificial intelligence paradigms for smart cyber-physical systems*, pages 1–19, 2021.

[3] Steffen Herbold, Alexander Trautsch, Benjamin Ledel, Alireza Aghamohammadi, Taher A Ghaleb, Kuljit Kaur Chahal, Tim Bossenmaier, Bhaveet Nagaria, Philip Makedonski, Matin Nili Ahmadabadi, et al. A fine-grained data set and analysis of tangling in bug fixing commits. *Empirical Software Engineering*, 27(6):125, 2022.

[4] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018.

[5] Great Expectations. Great expectations, 2025.

[6] City of Toronto. Dataset catalogue, 2025.