**📘 Research Document: Automated Data Quality Monitoring in Cloud Warehouses**

**🔖 Abstract**

As data becomes a critical asset in modern enterprises, ensuring its quality is crucial, especially in cloud-based data warehouses. This project proposes an **Automated Data Quality Monitoring System** that leverages **Gradio**, **Great Expectations**, and **Python** libraries to validate datasets uploaded in CSV format. The system provides a user-friendly dashboard to detect null values, uniqueness violations, duplicate rows, and row count anomalies. This facilitates early identification of data issues, improves reliability, and reduces time-to-insight for data-driven operations.

**🎯 Introduction**

With the rise of cloud data warehouses like Amazon Redshift, Google BigQuery, and Snowflake, large volumes of data are ingested and processed continuously. However, poor data quality—due to missing values, duplication, or incorrect formats—can severely impact business decisions and analytical outcomes. Existing data validation methods are often manual or require significant engineering effort. This project addresses this gap by offering an automated, interactive, and visual approach to data quality monitoring.

**🏗️ System Architecture**

**1. Frontend (User Interface)**

* **Gradio** is used to create a web-based interface.
* Users can upload CSV files and trigger analysis with a click.
* Results are displayed as:
  + Summary statistics (JSON)
  + Validation report (text)
  + Missing value chart (image)

**2. Backend Logic**

* Built using **Python** with **Pandas** and **Great Expectations**.
* Performs:
  + Null checks
  + Uniqueness check on the first column (assumed ID)
  + Row count range validation
  + Duplicate row detection
  + Missing values visualization using **Matplotlib** and **Seaborn**

**3. Port Configuration**

* Automatically detects and binds to an available port (between 7860–7960) using sockets.

**🔍 Methodology**

**Step-by-Step Flow:**

1. **Upload** CSV via dashboard.
2. **Read** file using pandas.read\_csv().
3. **Validate** with great\_expectations:
   * expect\_column\_values\_to\_not\_be\_null
   * expect\_column\_values\_to\_be\_unique
   * expect\_table\_row\_count\_to\_be\_between
4. **Visualize** missing data via bar chart.
5. **Summarize** analysis:
   * Number of rows/columns
   * Count of missing/duplicate entries
   * Timestamp
   * Pass/fail summary of all checks
6. **Display** results in real-time in the dashboard.

**🧪 Data Quality Metrics**

| **Metric** | **Purpose** |
| --- | --- |
| Null Value Check | Identify incomplete or missing records |
| Uniqueness Check | Ensure primary identifier columns are reliable |
| Row Count Validation | Detect partial file loads or unexpected truncation |
| Duplicate Detection | Catch redundant or mistakenly duplicated records |

**🧰 Tools & Libraries**

| **Tool** | **Role** |
| --- | --- |
| **Gradio** | UI for uploading files and showing results |
| **Pandas** | Data reading and manipulation |
| **Great Expectations** | Data validation framework |
| **Matplotlib & Seaborn** | Visualization of missing values |
| **Python Sockets** | Dynamic port assignment |
| **Logging Module** | Logs operational events and exceptions |

**💡 Use Cases**

* **Data Engineering Teams**: Quick verification before loading into data warehouses.
* **Business Analysts**: Ensure data integrity without technical expertise.
* **Machine Learning Pipelines**: Pre-checks before model training.
* **Cloud Integrations**: Plug into CI/CD workflows for automated validation.

**📈 Results (Sample Output)**

Upon uploading a sample CSV:

* The dashboard returned:
  + 1,200 rows, 12 columns
  + 50 missing values across 3 columns
  + 0 duplicate rows
  + 5 out of 5 validations passed ✅
  + Missing value chart showed clear gaps in two fields

**🔐 Security Considerations**

* Local-only web app (by default)
* No external data storage or cloud dependency
* Could be containerized and secured for enterprise use

**🚀 Future Scope**

| **Feature** | **Description** |
| --- | --- |
| Cloud Integration | Analyze data directly from S3, Azure Blob, etc. |
| Configurable Expectations | UI-driven rule setting per column/data type |
| Persistent Logs & Reports | Save results to database or cloud storage |
| User Authentication | Multi-user support with login and history |
| Real-time Monitoring | Trigger alerts on incoming pipeline data |

**📚 References**

* Great Expectations Documentation
* [Gradio Documentation](https://www.gradio.app/)
* Python Libraries: pandas, matplotlib, seaborn

**📎 Appendix**

**Installation**

bash

CopyEdit

pip install -r requirements.txt

**Running the App**

bash

CopyEdit

python data\_quality\_dashboard.py

Then open your browser at the suggested URL (e.g., <http://127.0.0.1:7860>)