# **Bigspark Data Quality Guardrail: Integration Guide for RAI Compliance**

## **Overview**

The **Bigspark Data Quality Guardrail Starter Pack** is a modular, Python-native validation suite designed to assess and score datasets against six core data quality dimensions as recognized in responsible AI (RAI) governance:

* **Accuracy**
* **Completeness**
* **Consistency**
* **Uniqueness**
* **Validity**
* **Timeliness**

This package is developed to align with regulatory frameworks **(e.g. EU AI Act, ISO/IEC 42001, FCA AI guidance)**, enabling integration into RAI pipelines and sandbox environments.

## **Why It Matters**

Responsible AI requires **trustworthy data**. Before AI models are trained or deployed, organizations must verify that data meets minimum quality and compliance thresholds. The Bigspark DQ Guardrail helps ensure:

* Clean, structured, and trustworthy inputs
* Repeatable audit-ready quality assessments
* Automated pre- and post-deployment checks
* Compliance with legal and regulatory frameworks
* Compatibility with the **RAI Metadata Scoring Model (1–5)**

## **Key Features**

| **Feature** | **Description** |
| --- | --- |
| **Feature** | **Description** |
| **Python-native** | No external cloud dependencies; full control over code |
| **Six DQ dimensions** | Test suite covers all foundational data quality metrics |
| **YAML-configurable** | Easy threshold tuning and schema definition |
| **JSON output for scoring** | Scores, status, issues, and metadata structured for RAI pipelines |
| **Compatible with RAI rating** | Ratings are mapped to 1–5 scale used in FCA/NayaOne rai-core framework |
| **Modular design** | Can be extended with Fairlearn, AI Fairness 360, or Evidently (optional) |

## 

## **Package Structure**

bigspark-data-quality-guardrails/

├── data\_quality/

│ ├── accuracy.py

│ ├── completeness.py

│ ├── consistency.py

│ ├── uniqueness.py

│ ├── validity.py

│ ├── timeliness.py

│ └── dq\_runner.py

├── configs/

│ └── thresholds.yaml

├── tests/

│ └── test\_runner.py

├── requirements.txt

└── README.md



## **How It Works**

1. **Load Dataset**: Any pandas DataFrame is supported (e.g., raw CSV, fraud training data, experimental sandbox).
2. **Define Config**: Use thresholds.yaml to specify:  
   * Expected schema types
   * Thresholds for quality
   * Key columns for uniqueness/timeliness
3. **Run DQ Tests**:

from data\_quality.dq\_runner import run\_all\_dq\_tests

results = run\_all\_dq\_tests(df, config)

1. **Get Output**:  
     
    Each test returns:

{

"dimension": "accuracy",

"score": 0.98,

"status": "pass",

"rating": 3,

"issues": [],

"metadata": { ... }

}



## **Integration into RAI Core Repository**

You can integrate this into an RAI pipeline as follows:

### **✅ 1. Plug into the RAI metadata scoring process**

Each rating returned by a DQ test maps directly to RAI’s 1–5 scale. You can embed this into the metadata declaration YAML:

data\_quality:

accuracy: 3

completeness: 2

validity: 3

timeliness: 1



Or, use the results as part of a compliance assertion during model registration.

### **✅ 2. Use in Sandbox (NayaOne or Digital Sandbox)**

Add the test runner as a pre-processing container in:

* **Data Preparation** stage
* **Pre-deployment Testing** pipeline
* Or even **continuous validation jobs**

NayaOne can support this via:

* Python container execution
* API-based scoring (if wrapped with FastAPI)

## **How to Use It in a Project**

### **Step 1: Set Up Your Project**

Clone the repository or copy the data\_quality/, tests/, configs/, and data/ folders into your project:

git clone https://github.com/your-org/bigspark-data-quality-guardrails.git

cd bigspark-data-quality-guardrailsTo get started with the Bigspark Data Quality Guardrail, clone the repository:

git clone https://github.com/your-org/bigspark-data-quality-guardrails.git

Then, navigate into the cloned directory:

cd bigspark-data-quality-guardrails



(Optional) Create and activate a virtual environment:

python -m venv dq\_env

source dq\_env/bin/activate

pip install -r requirements.txt



### **Step 2: Prepare Your Input Files**

* Place your dataset in the data/ folder as input\_data.csv
* Configure thresholds and schema in configs/thresholds.yaml

data/input\_data.csv

configs/thresholds.yaml



### **Step 3: Run the DQ Tests**

Use the provided test runner to validate your data:

PYTHONPATH=. python tests/test\_runner.py

PYTHONPATH=. python tests/test\_runner.py



This will:

* Run data quality checks
* Print a summary to the console
* Save results to reports/dq\_report.json

### **Step 4: Use the Output**

Your JSON report will look like:

[

{

"dimension": "accuracy",

"score": 0.96,

"status": "pass",

"rating": 3,

"issues": []

},

...

]



You can use this output to:

* Feed into RAI compliance dashboards
* Track metadata lineage
* Inform release decisions and QA gates

**Extending with Optional Tools**

If desired, extend the DQ framework using:

* **Evidently AI** (for dashboard-based drift tracking)
* **AI Fairness 360** (for fairness bias testing)
* **Great Expectations** (for test-driven data contracts)

These are not required for baseline guardrail testing, but offer enhanced observability and auditability.

## **Conclusion**

The Bigspark DQ Guardrail Package is ready to serve as a standardized, auditable, and extensible module for AI experimentation and compliance readiness. It supports integration into sandbox environments, model governance tools, and production AI pipelines with minimal overhead.

It enables FinTechs, regulators, and platform providers (e.g., NayaOne, FCA, etc.) to align on clear, measurable, and portable standards for ensuring **trustworthy data for trustworthy AI**.