DQ-AI: Cross-Platform Intelligent Data Quality for Heterogeneous Systems (Nordea Credit Risk)

One-line: A platform-independent, AI-assisted engine that profiles any dataset, flags issues proactively, learns constraints, detects anomalies and drift, estimates uncertainty, explains root causes via lineage, and auto-generates quality checks as code.

---

1) Executive Summary

- **Problem**: Data quality across Hadoop/Oracle/Snowflake/... is reactive, manual, and brittle.
- **Solution**: **DQ-AI**: a metadata-first system that connects to many backends, builds statistical and semantic profiles, learns rules from "golden" data, surfaces anomalies with uncertainty, and **automatically proposes (and optionally applies) quality controls**.
- **Differentiators**
- **AI-driven suggestions**: LLM + constraint-learning to synthesize checks in a **DQ-DSL** or Great Expectations (GE) YAML.
- **Uncertainty maps**: quantify where imputation/measurement is shaky; surface low-confidence regions.
- **Causal-ish RCA**: lineage-aware blame assignment using graph traversal + change-point + contribution scores.
- **Platform independence**: connector model using SQLAlchemy/JDBC + Arrow; OpenLineage for lineage; deployable on Airflow/DBT/Databricks.
- **Risk domain packs**: credit-risk-specific rules (e.g., sentinel values, age/tenor constraints, PD/LGD distributions).

Demo target (Kaggle Home Credit Default Risk): load tables, auto-profile, flag sentinel values (e.g., `DAYS_EMPLOYED==365243`), duplicates on `SK_ID_CURR`, outliers in `EXT_SOURCE_*`, drift vs. training baseline, suggest checks, run RCA on failing batches.

---

2) High-Level Architecture

```
                    ┌──────────────────────────────┐
    Sources         │  Connectors & Ingestion      │  JDBC/ODBC, S3/Parquet,
HDFS,
 (Oracle, S3,       │  (Batch & Streaming)         │  Snowflake, BigQuery,
Kafka
  Snowflake, ...)   └──────────────┬───────────────┘
                                   │ Arrow/Parquet frames
                                   ▼
                          ┌───────────────┐
                          │  Profiler     │  (schema, stats, distributions,
                          │               │   embeddings, PII tags)
                          └──────┬────────┘
                                 │ profiles & samples
```

```
                   ▼
          ┌───────────────────────────┐
          │ AI Suggestion Service     │  (LLM + rule templates +
          │  & Constraint Learner     │   metric search)
          └───────────┬───────────────┘
                      │ DQ-DSL / GE YAML
                      ▼
          ┌───────────────────────────┐
          │ Check Runner              │  (spark/sql/pandas backends)
          └───────────┬───────────────┘
                      │ results (pass/fail, scores)
                      ▼
       ┌───────────────────────────────┐
       │ Anomaly & Drift Detector      │  (IForest, KS/PSI/JS, CPD)
       └──────────┬────────────────────┘
                  │ anomalies with uncertainty
                  ▼
        ┌─────────────────────────────┐
        │ Lineage & RCA Graph         │  (OpenLineage + graph DB)
        └─────────┬───────────────────┘
                  │ explanations, blast radius
                  ▼
           ┌────────────────────┐
           │ Command Center UI  │  (dashboards, chat-ops,
           │ & APIs             │   approval workflows)
           └────────────────────┘
```

Storage: Postgres (metadata), object store (artifacts), **graph DB** (Neo4j/JanusGraph) for lineage, vector index (FAISS) for column embeddings.

Execution: Containerized microservices + jobs on Spark/Databricks or Pandas/DuckDB depending on scale. Orchestrate via Airflow/DBT/Argo.

---

3) Data Model (Core Tables)

- **datasets**(id, name, platform, location, owner, pii_level)
- **columns**(id, dataset_id, name, dtype, semantic_tag, embedding)
- **profiles**(dataset_id, version, stats_json, ts)
- **checks**(id, dataset_id, dsl_yaml, level, source = {template|ai|manual})
- **runs**(id, dataset_id, check_id, status, score, sample_uri, ts)
- **anomalies**(id, dataset_id, type, region_spec, severity, uncertainty, evidence, ts)
- **lineage_nodes**(id, type={source, transform, dataset, model}, attrs)
- **lineage_edges**(src_id, dst_id, op, attrs)
- **rca_findings**(id, anomaly_id, suspect_node_id, blame_score, rationale)

---

4) Profiling & Semantics

What we compute - **Schema**: types, nullability, uniqueness, referential links. - **Stats**: counts, nulls, distincts, quantiles, histograms, correlations, time seasonality. - **PII detection**: regex + ML classifier (IBAN, SSN, phone). - **Embeddings**: column-name + sample-value text embedding to capture semantics (e.g., `DAYS_BIRTH` ≈ "customer age").

Output: Profile JSON + small sampled rows; cached by data version (e.g., LakeFS/Delta log hash or snapshot timestamp).

---

5) AI-Driven Suggestions

Two engines working together: 1) **Template Retrieval** using embeddings - Query vector = [column name, description, example values, stats]. - Retrieve rule templates: e.g., *monotone time*, *bounded age [18, 100]*, *valid enums*, *no sentinel 365243*, *foreign-key integrity*. 2) **LLM Synthesis** into **DQ-DSL** or **Great Expectations YAML** - Converts suggestions into executable checks with thresholds tuned to profiles (e.g., `null_ratio < 0.5%`, `psi < 0.2`).

Example DQ-DSL (compiles to GE)

```
checks:
  - name: days_employed_not_sentinel
    when: dataset == "application_train"
    rule: column("DAYS_EMPLOYED").not_in([365243])
    severity: high
  - name: age_bounds
    rule: between(column("DAYS_BIRTH") / -365.25, 18, 100)
  - name: ext_source_range
    rule: all(columns ~ "^EXT_SOURCE_").between(0,1)
  - name: duplicates_on_sk
    rule: unique(["SK_ID_CURR"])
```

Active learning: user accepts/edits; feedback updates retrieval weights and thresholds.

---

6) Anomaly & Drift Detection

- **Point/Row anomalies**: Isolation Forest, Local Outlier Factor on normalized numeric + encoded categoricals.
- **Distribution drift**: Kolmogorov–Smirnov, Jensen–Shannon, Population Stability Index (PSI), Cramér–von Mises for univariate; MMD for multivariate.
- **Time series**: STL decomposition + residual thresholds; change-point (Bayesian/ruptures) for metrics.
- **Context-aware**: stratify by country/product/segment; compare within peers not global.

Severity = impact × prevalence × business weight (e.g., features used in PD model get higher weight).

---

7) Uncertainty Identification

We provide **uncertainty maps** at three levels: 1) **Imputation uncertainty**: run multiple imputation (e.g., MICE) and compute per-cell variance; mark high-variance cells. 2) **Model uncertainty** (if downstream PD/LGD scoring is available): conformal prediction intervals or entropy of class probs; flag rows where predictions are unstable to small perturbations. 3) **Measurement uncertainty**: cross-system reconciliation (same entity, different source) → disagreements get a confidence score via consensus/credibility weighting.

Output: per-row/column **uncertainty score** (0–1) used to sort review queues and to attenuate anomaly severity when confidence is low.

---

8) Data Lineage & Root Cause Analysis (RCA)

- **Capture** lineage via **OpenLineage** events emitted from Airflow/DBT/Spark jobs (source, transformation, output). Store in graph DB.
- **Augment** with column-level mappings where available (DBT exposures, Spark plan lineage).
- **RCA algorithm**
- For an anomaly on dataset D at time t, find **upstream frontier** within the last N hops.
- Compute **metric deltas** (nulls, distincts, value ranges) for each node around t; run **change-point detection**.
- Score suspects by (a) temporal alignment, (b) dependency proximity, (c) historical failure correlation, (d) **blast radius** (how many downstream datasets affected).
- Produce **actionable hypothesis**: e.g., "Null spike in `EXT_SOURCE_3` originates in `landing.raw_ext3` since 2025-09-12; job `ingest_ext3` changed schema; fallback default set to NULL."

OpenLineage event (sketch)

```
{
  "job": {"name": "dbt.run.application_train"},
  "inputs": [{"namespace":"snowflake://risk.raw","name":"application_raw"}],
  "outputs": [{"namespace":"snowflake://
risk.marts","name":"application_train"}],
  "run": {"runId": "..."},
  "facets": {"schema": {"fields": [{"name":"DAYS_BIRTH","type":"INTEGER"}]}}
}
```

---

9) Platform Independence

- **Connectors**: SQLAlchemy/JDBC; filesystem (S3/GCS/ADLS/HDFS), Kafka; Pandas/DuckDB for local; Spark for scale.
- **Execution abstraction**: checks compile to **SQL** or **Spark** or **Pandas** via adapter layer.
- **Data interchange**: Apache **Arrow**; sampling pushed down with predicates.
- **Deployment**: Helm charts; runs on-prem or cloud; secrets via Vault/KMS.

---

10) Prototype Plan (Hackathon-ready)

Scope (2 days): 1) **Ingest & Profile**: load Kaggle `application_train.csv` (and one aux table), compute profiles & embeddings; PII scan off by default. 2) **AI Suggestions**: retrieve templates + synthesize 6–10 checks into GE YAML/DQ-DSL. 3) **Run Checks**: execute with Pandas backend; show pass/fail with samples. 4) **Anomaly/Drift**: IsolationForest for row anomalies; PSI/KS vs. a baseline split. 5) **Uncertainty**: simple MICE imputation variance on 2–3 key columns. 6) **Lineage+RCA (mock)**: static mini-graph (raw → cleaned → train) and RCA scoring demo. 7) **UI**: Streamlit/Gradio "Command Center" + lineage mini-graph (pyvis) + chat-ops for “suggest checks for dataset X”.

Stretch: DBT/Great Expectations integration; OpenLineage events; basic Snowflake connector (read-only).

---

11) Sample Notebook Snippets (Pandas-scale)

> Replace `DATA_DIR` with your path to the Kaggle dataset.

11.1 Profiling & Basic Issues

```
import pandas as pd, numpy as np
from scipy import stats
DATA_DIR = "/path/to/home-credit-default-risk"
X = pd.read_csv(f"{DATA_DIR}/application_train.csv")

profile = {
  "rows": len(X),
  "nulls": X.isna().sum().to_dict(),
  "distinct": {c: X[c].nunique(dropna=True) for c in X.columns},
  "quantiles": X.quantile([0.01,0.05,0.5,0.95,0.99],
numeric_only=True).T.to_dict()
}

issues = []
if (X['DAYS_EMPLOYED']==365243).any():
    issues.append({"col":"DAYS_EMPLOYED","type":"sentinel","value":365243})
if (X['DAYS_BIRTH'] > 0).any():
    issues.append({"col":"DAYS_BIRTH","type":"sign_error"})
if X.duplicated(subset=['SK_ID_CURR']).any():
    issues.append({"type":"duplicate_key","key":"SK_ID_CURR"})
```

11.2 Distribution Drift & PSI

```
def psi(a, b, bins=10):
    qa, qb = np.quantile(a.dropna(), np.linspace(0,1,bins+1)),
np.quantile(b.dropna(), np.linspace(0,1,bins+1))
    qa[0]=qb[0]=min(qa[0],qb[0]); qa[-1]=qb[-1]=max(qa[-1],qb[-1])
    ha,_ = np.histogram(a, bins=qa); hb,_ = np.histogram(b, bins=qb)
```

```
    pa = (ha + 0.5) / (ha.sum() + 0.5*bins)
    pb = (hb + 0.5) / (hb.sum() + 0.5*bins)
    return ((pa - pb) * np.log(pa/pb)).sum()

base = X.sample(frac=0.5, random_state=42)
new  = X.drop(base.index)
psi_ext1 = psi(base['EXT_SOURCE_1'], new['EXT_SOURCE_1'])
```

11.3 Row Anomalies (Isolation Forest)

```
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
num =
X.select_dtypes(include=[np.number]).fillna(X.select_dtypes(include=[np.number]).median())
Z = StandardScaler().fit_transform(num.values)
iso = IsolationForest(n_estimators=200, contamination=0.02, random_state=0)
score = -iso.fit_predict(Z)  # 2 for outlier, 1 for inlier → convert to {0,1}
X['anomaly_score'] = (score == 2).astype(int)
```

11.4 Uncertainty via Multiple Imputation Variance

```
# Toy: multiple imputations via random forests or simple stochastic
imputation
M = 5
imputations = []
for m in range(M):
    imp = num.copy()
    for c in imp.columns:
        if imp[c].isna().any():
            col = imp[c]
            mu, sd = col.mean(), col.std(ddof=1)
            imp.loc[col.isna(), c] = np.random.normal(mu, sd,
size=col.isna().sum())
    imputations.append(imp)
stack = np.stack([imp.values for imp in imputations])
cell_var = stack.var(axis=0)
uncertainty = cell_var.mean(axis=1)
X['uncertainty_row'] = uncertainty
```

11.5 Suggestion Synthesis (template → GE YAML)

```
suggestions = [
  {"name":"days_employed_not_sentinel","ge":{
     "expect_column_values_to_not_be_in_set":{
       "column":"DAYS_EMPLOYED","value_set":[365243]}}},
  {"name":"age_bounds","ge":{
     "expect_column_values_to_be_between":{
```

```
"column":"DAYS_BIRTH","min_value":-365.25*100,"max_value":-365.25*18}}},
  {"name":"ext_source_range","ge":{
     "expect_column_values_to_be_between":{
       "column":"EXT_SOURCE_3","min_value":0,"max_value":1}}}
]
```

---

12) APIs (sketch)

- `POST /profile` {connection, dataset} → profile JSON
- `POST /suggest` {profile} → DQ-DSL/GE YAML + rationales
- `POST /run_checks` {dataset, checks} → results + samples
- `POST /detect_anomalies` {dataset, baseline?} → anomalies + uncertainty
- `GET /lineage/:dataset` → nodes/edges
- `POST /rca` {dataset, anomaly_id} → suspect nodes + blame scores

---

13) UI: Data Quality Command Center

- **Overview**: score by dataset; trend of failures; MTTD/MTTR.
- **Dataset view**: profile cards; check list; fail samples; **uncertainty heatmap**; drift charts.
- **Lineage**: interactive graph; click to see metrics; **"Explain"** button runs RCA.
- **Chat-Ops**: "Suggest checks for application_train" → renders YAML, with **Apply** (PR to repo) or **Schedule** (Airflow).

---

14) Security & Governance

- **RBAC** integrated with IdP; project-level permissions.
- **PII handling**: masking, tokenization; opt-in profiling of sensitive fields; field-level audit logs.
- **On-prem friendly**: all components containerized; outbound egress optional; model weights local.

---

15) Evaluation & Success Metrics

- **Precision/Recall of anomaly flags** vs. labeled subset.
- **PSI/KS improvements** after rules applied.
- **MTTD/MTTR** reduction in incidents.
- **Adoption**: % datasets with auto-generated checks accepted; review time saved.

---

16) Demo Script (10 minutes)

1) Connect CSV/Snowflake (read-only). Auto-profile appears. 2) Click **Suggest checks** → show YAML + rationales. 3) **Run** → failures for sentinel/duplicates; show samples. 4) Trigger drift by swapping split →

PSI warning with uncertainty bands. 5) Open **Lineage** → run RCA → surface suspect ingest job. 6) **Apply** → commit GE YAML to repo; schedule daily run.

---

17) Roadmap

- Week 1–2: harden connectors; DBT/GE integration; OpenLineage events.
- Week 3–4: active learning loop; column-level lineage extraction for Spark/DBT.
- Week 5+: causal discovery experiments; auto-repair suggestions; policy enforcement gates (CI/CD for data).

---

18) Why this wins (Nordea context)

- Targets **credit risk** realities (sentinels, external scores, aging features).
- **Explains** incidents to model/governance stakeholders (RCA + lineage).
- **Portable** across on-prem/cloud estates.
- Moves teams from reactive triage to **proactive, self-tuning** DQ management.

Appendix A: Minimal Lineage Graph (for demo)

```
[raw.ext_sources] --(ingest_ext3)--> [staging.ext3] --(clean_ext3)--> 
[marts.application_train]

                                                --(join_customer)--> 
[marts.customer_features]
```

Appendix B: Domain-Specific Rule Ideas (Credit Risk)

- `AMT_INCOME_TOTAL` within 5× IQR per country; winsorize else flag.
- `DAYS_BIRTH` ∈ [−365*100, −365*18].
- Exposure/tenor monotonicity checks by product.
- **Cross-table**: `SK_ID_CURR` must exist in master; no orphan records in history tables.

Appendix C: Check Severity Levels

- **High**: key uniqueness, referential integrity, PII leakage, schema change.
- **Medium**: outliers beyond robust limits, drift PSI 0.2–0.4.
- **Low**: formatting/casing/style.

Prepared for: Nordea Bank Finland Abp — Group Risk (Risk Models)