

• Data Trust & Anomaly Detection Platform

A real-time Data Quality and ML-driven anomaly monitoring system built on synthetic customer acquisition data.

1. Overview

This project builds an end-to-end framework for monitoring the health of a customer acquisition data pipeline. It validates incoming datasets using both Technical Data Quality (TDQ) and Business Data Quality (BDQ) rules, and then applies machine learning-based anomaly detection to uncover subtle drifts and irregularities that traditional rule-based systems often miss.

The solution follows patterns commonly used by modern data engineering and AI teams to monitor 100+ real-time tables. All datasets used here are synthetic, but they reflect realistic patterns across the customer journey — starting from a visitor landing on the platform and progressing through applications, account openings, and transaction activity.

2. Dataset Description (Synthetic)

The project operates on five synthetic datasets:

- **visitor_events.csv** – visitors landing on site
- **marketing_source.csv** – campaigns and attribution
- **applications.csv** – applications submitted
- **accounts.csv** – approved accounts
- **transactions.csv** – customer transaction activity

Why synthetic?

- It avoids any privacy concerns
- It allows injecting controlled errors and anomalies
- It supports reproducible evaluation
- It follows the same structure as real customer acquisition funnels (user → application → account → activity).

These tables include intentionally created issues like inconsistent mappings, missing values, timestamp gaps, unrealistic ranges, and transaction spikes — all meant to test whether the system catches them.

3. Project Structure

```
data/
    raw + cleaned datasets
    tdq_raw.csv
    tdq_clean.csv
    bdq_report.csv
    reports/anomaly/*.csv

src/
    cleaning_functions.py
    tdq/
        tdq_checks.py
        run_tdq.py
    bdq/
        bdq_checks.py
        run_bdq.py
    anomaly/
        anamoly_pipeline.py
    visuals.py

notebook/
    tdq_visualizations.ipynb
    bdq_visualizations.ipynb

visuals/
```

4. Technical Data Quality (TDQ)

The TDQ layer confirms that each dataset satisfies foundational expectations before business logic is applied.

The checks include:

- **Schema Validation**

Ensures column order, datatypes, and required fields match the expected schema.

- **Completeness**

Flags missing values in key identifiers, timestamps, and categorical fields.

- **Uniqueness**

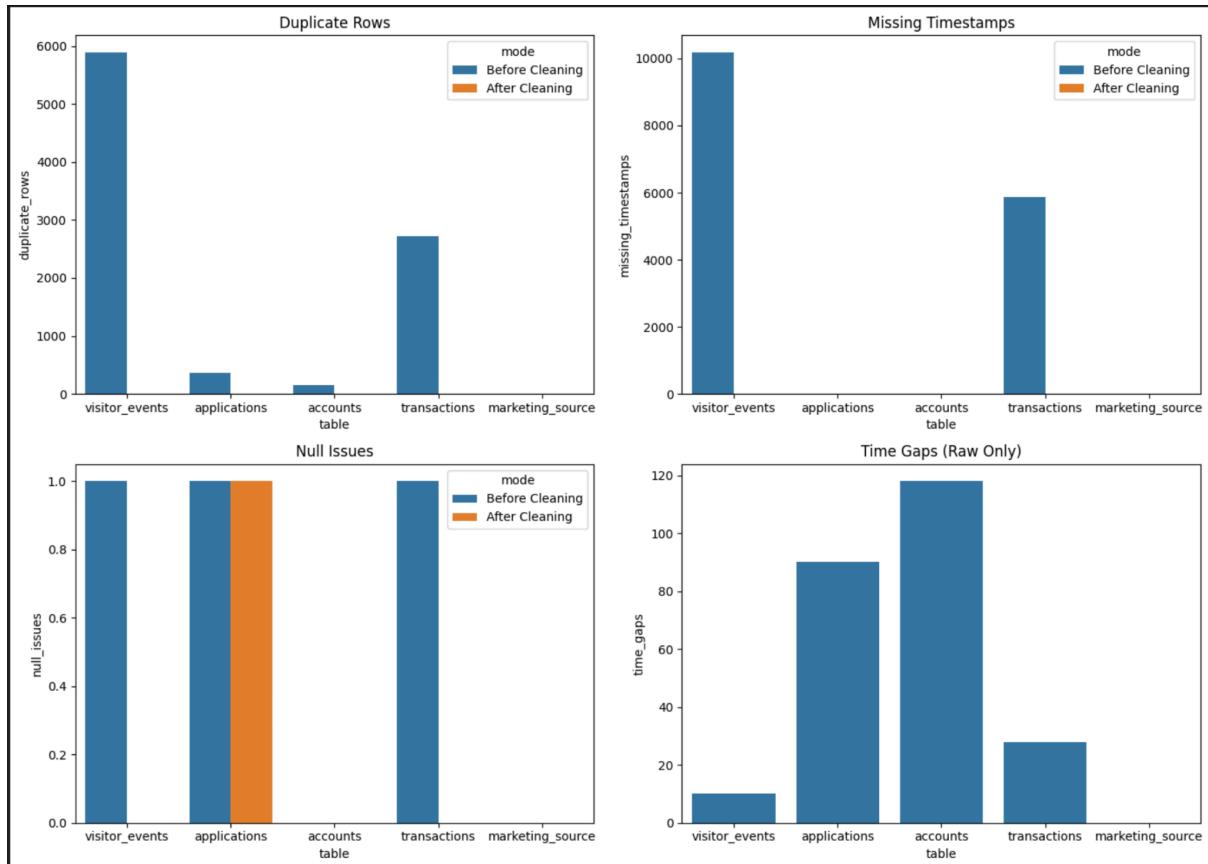
Identifies duplicated rows or repeated primary keys.

- **Freshness**

Detects delays in data ingestion or stale partitions.

- **Volume Drift**

Compare row counts to historical baselines to catch sudden spikes or drops.



5. Business Data Quality (BDQ)

BDQ validates **business logic** and **funnel integrity**, ensuring that lifecycle relationships across tables remain intact.

- **Visitor → Application Mapping**

Every application must tie back to a valid visitor.

- **Application → Account Mapping**

Accounts should originate from an existing application.

- **Marketing Attribution**

Ensures correct linkage between campaign → source → visitor.

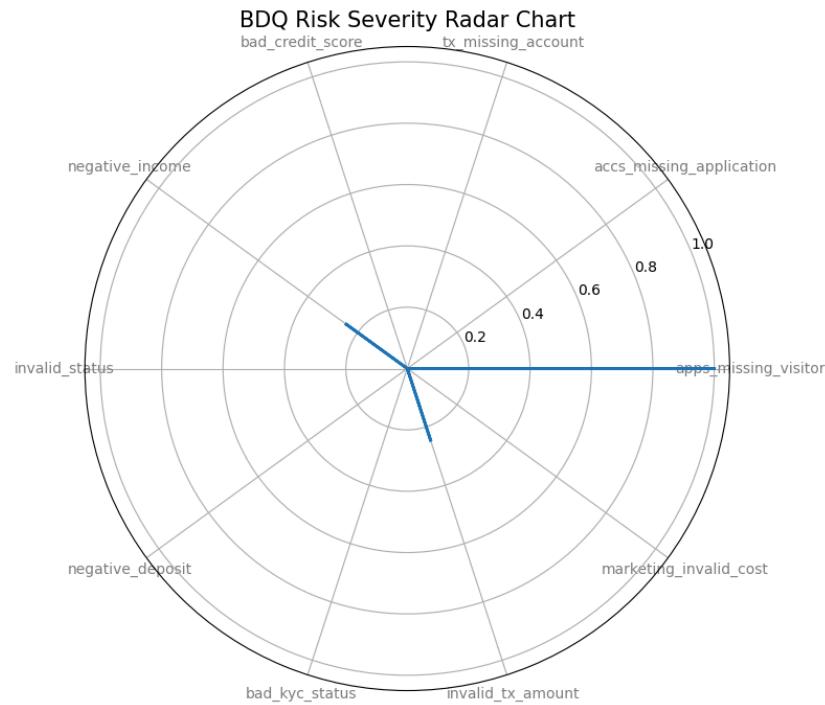
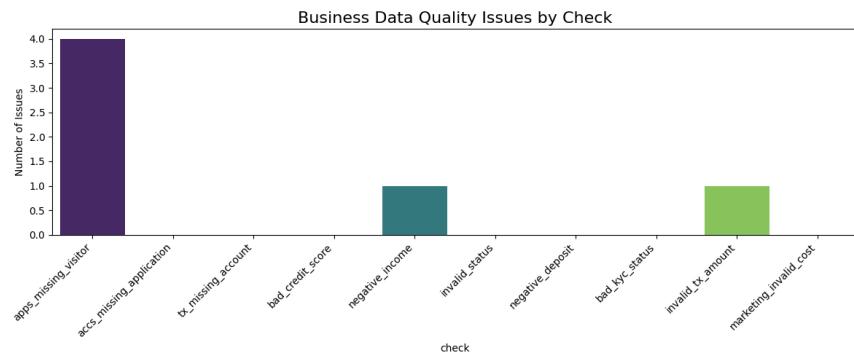
- **Range Checks**

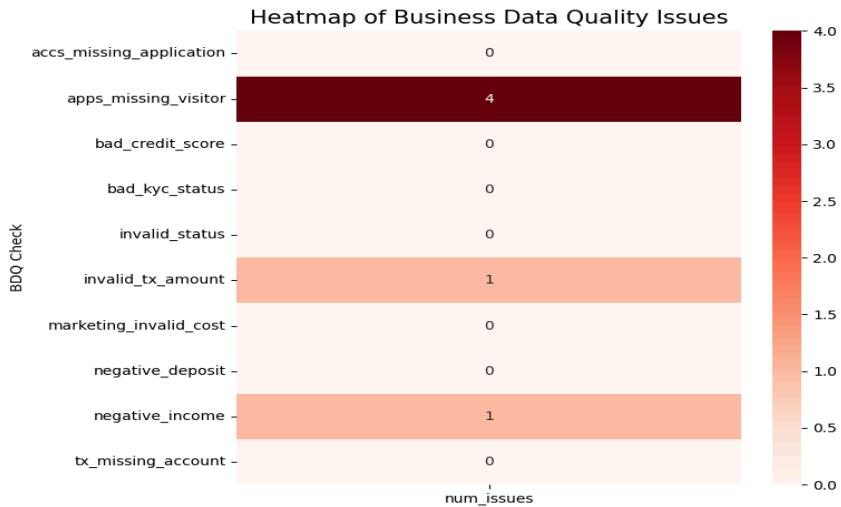
Flags invalid values (e.g., negative transaction amounts, unrealistic ages).

- **Cross-Table Consistency**

Confirms ID alignment and lifecycle progression.

BDQ results are stored in `bdq_report.csv`.





6. ML-Based Anomaly Detection

This step detects time-series anomalies in aggregate metrics as well as row-level anomalies in user activity.

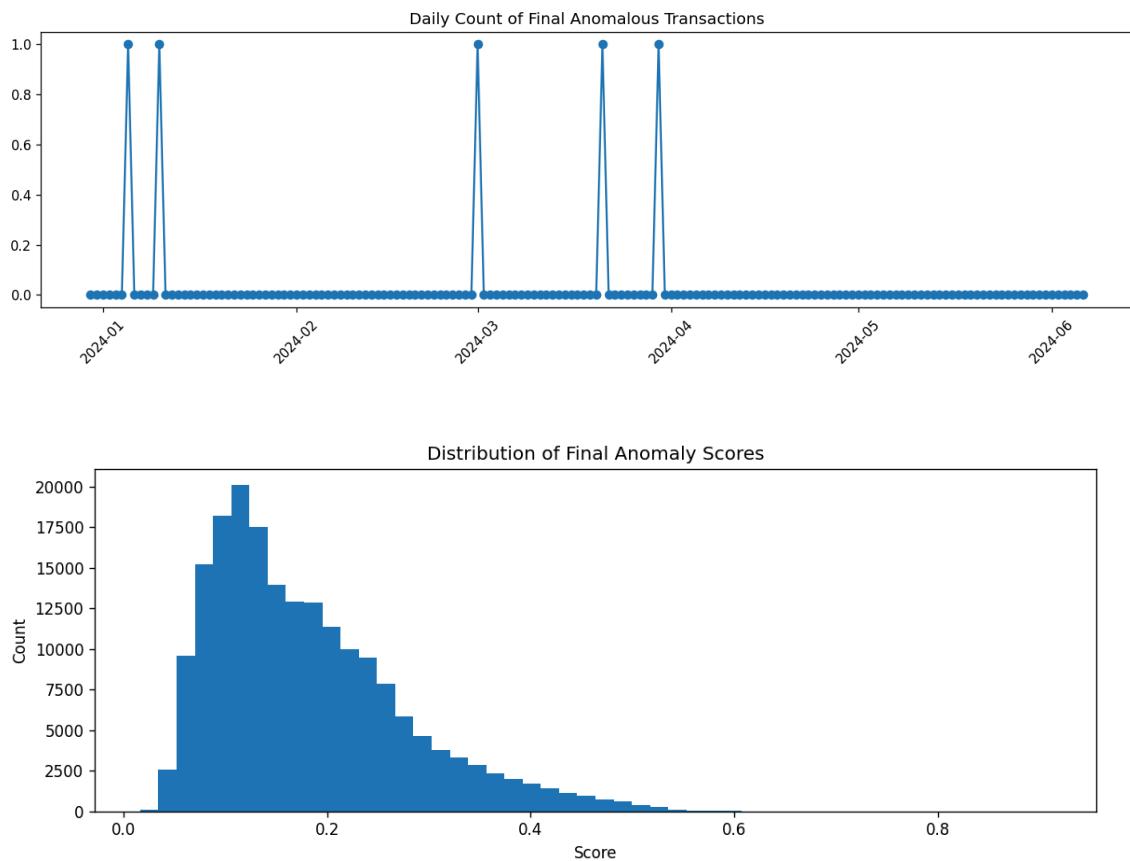
A. Time-Series Anomaly Detection

Applied to:

- Hourly visitor counts
- Application submissions
- Account creation
- Transaction volume
- Conversion ratios

Techniques used:

- Isolation Forest
- Rolling Z-score
- Seasonal decomposition
- Trend-sensitive anomaly scoring



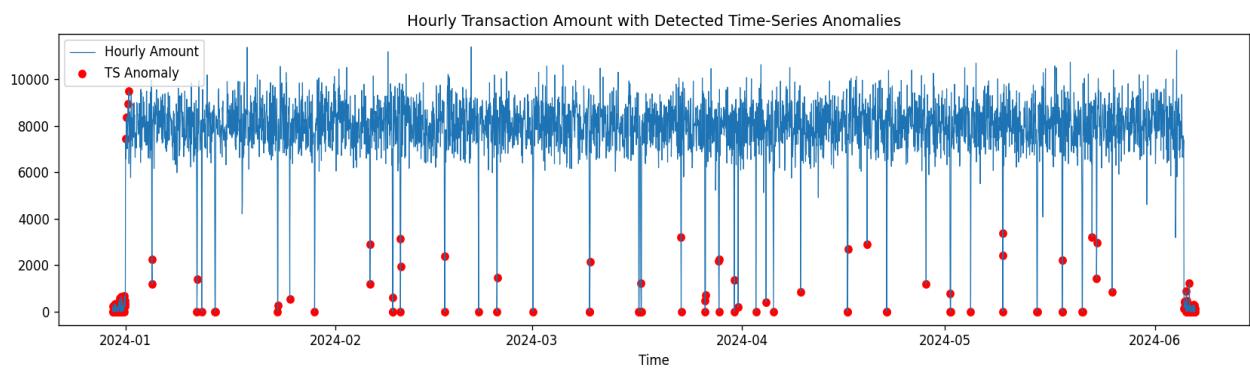
B. Row-Level Anomaly Detection

Used primarily for the **transactions** table.

The model identifies unusual behaviors that may indicate fraud, suspicious activity, or system errors.

Model used: **Isolation Forest**, operating on fields such as:

- amount
- risk_score
- velocity
- user/device identifiers
- timestamps



7. Results Summary

TDQ Findings

Based on the checks across all five tables:

- **visitor_events** contained 1 null issue, **5,890 duplicate rows**, and **10 timestamp gaps**.
- **applications** had 1 null issue, **359 duplicates**, 2 datatype mismatches, and **90 timestamp gaps**.
- **accounts** showed **145 duplicate rows**, 1 datatype mismatch, and **118 timestamp gaps**.
- **transactions** contained 1 null issue, **2,720 duplicates**, **5,880 missing timestamps**, and **28 timestamp gaps**.
- **marketing_source** passed all TDQ checks with no major issues.

These checks confirm that while schemas were aligned, the synthetic data included intentional quality problems such as duplicates, missing timestamps, and minor type mismatches—most of which were corrected in the cleaned outputs.

BDQ Findings

From the BDQ layer:

- **4 applications** did not map back to a valid visitor.
- No accounts or transactions were missing their required upstream references.
- **1 record** was flagged for a negative income value.

These issues reflect funnel inconsistencies that the BDQ layer is designed to detect.

Anomaly Detection Findings

Across the ML-driven anomaly modules:

- The time-series model detected **164 hourly anomalies** in transaction volume and amount patterns.
- The row-level Isolation Forest model identified **193,746 transaction-level anomalies**.

Overall, the anomaly layer successfully surfaced both structural and behavioral irregularities across the funnel.

8. Conclusion

This project builds a complete Data Trust Layer for monitoring the quality and stability of a customer acquisition pipeline. It combines foundational TDQ checks, business logic validation, and ML-based anomaly detection to catch a wide range of issues—from duplicate

records and missing timestamps to inconsistent mappings and abnormal transaction behavior.

Even though the datasets are synthetic, they mimic real-world data challenges and demonstrate how the system performs under practical conditions. The modular design makes it easy to extend: more tables, additional checks, or even streaming pipelines can be added without major changes to the overall framework. As data volume grows, the pipeline can be scaled on platforms like Spark, BigQuery, or AWS, making it suitable for production-level environments and continuous monitoring.