Barclays
Hack-O-Hire

# Anomaly Detection System

# PRESENTED BY

Data Dragons

| | |
|---|---|
| **Ananya Kinha** | RA2111026010332 |
| **Yashwardhan Khanna** | RA2111026010338 |
| **Ashish Kumar Srivastava** | RA2111026010317 |
| **Johan Mathew Joseph** | RA2111026010309 |

ABSTRACT

# An executive summary.

As the banking sector increases its reliance on **digital transactions**, the precision of transaction settlement processes becomes critical. These settlements depend on sourced, transformed, and standardized data from various systems. Despite handling **millions of records daily**, the system is susceptible to anomalies such as incorrect pricing, data entry errors, and fraudulent activities, potentially leading to substantial financial discrepancies. The proposed Anomaly Detection System (ADS) is designed to identify and alert **suspicious transactions in real-time**.

Our Anomaly Detection System distinguishes itself through **advanced machine-learning algorithms** and **real-time processing capabilities**. By employing **Python** for its robust libraries and **AWS services** for its seamless integration, database management systems, ETL, and processing, our system ensures a high detection rate of anomalous transactions with minimal false positives. This combination of technologies allows for the agile development of a **scalable, secure, and efficient** anomaly detection platform.

One of the unique aspects of the ADS is its **emphasis on data security** and privacy. Recognizing the sensitivity of financial data, the system incorporates **end-to-end encryption**, leveraging **AWS KMS** for key management and ensuring that all data in transit and at rest is securely encrypted. Access controls and regular audits further enhance the system's security posture, safeguarding against unauthorized access and data breaches. Our Anomaly Detection System, augmented by **Tableau's visualization** capabilities, marks a significant advancement in any organization's ability to identify and mitigate fraud effectively.

Our Anomaly Detection System, powered by machine learning algorithms, is built to *streamline anomaly detection*. It prioritizes data security, leveraging encryption to protect sensitive information. The system highlights why transactions are flagged, aiding swift analysis and decision-making. Focusing on these elements effectively reduces anomaly rates and enhances an organization's security posture.

# Table of Contents

Problem Statement and Motivation 01

Dataset 02

Technology Stack 03

Detailed System Design 04

Implementation 06

Conclusion 08

# Problem Statement

- With the ever-growing volume of financial transactions, **millions of data points** are processed daily, sourced from a number of unique systems. This data, essential for accurate transaction settlements, is **susceptible to a wide range of anomalies** ranging from minor inaccuracies to significant errors like inflated values or data omissions.
- These issues are compounded by the growing threat of **payment frauds, scams, erroneous calculations, and missing bills**, which introduce additional complexity and urgency to detect and correct such anomalies.
- The challenge is to efficiently identify and rectify these irregularities and anomalous activities in real time, ensuring the **integrity and reliability** of the financial transaction process.

An **effective Anomaly Detection Framework** is crucial for mitigating these risks, by proactively spotting and addressing deviations from normative data patterns.

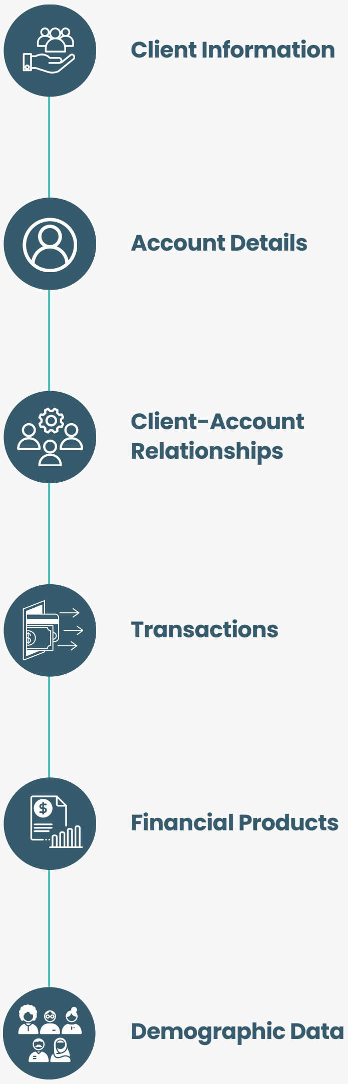
# Motivation

The drive to develop an ADS (Anomaly Detection System) is fueled by the necessity to:

- Preserve the **accuracy and integrity** of financial transactions against operational errors and anomalous schemes.
- Enhance the **security measures** to detect and prevent scams and frauds, protecting financial assets and customer trust.
- Uphold the financial ecosystem's **reliability**, ensuring smooth, secure, and trustworthy operations for all stakeholders.

# Dataset

- This system should utilize a dataset structure similar to the one stated below.
- We will be using an up-to-date version of the Czech financial dataset encompassing various aspects of a bank's operations. The data covers:
  - **Client Information:** Details about clients associated with the bank (relation: fin_client).
  - **Account Details:** Static characteristics of accounts, including creation date and branch address (relation: fin_account).
  - **Client-Account Relationships:** Links between clients and their corresponding accounts (relation: fin_disposition).
  - **Transactions:** Details of financial transactions occurring on accounts (relations: fin_order, fin_transaction).
  - **Financial Products:** Information on loans granted and credit cards issued (relations: fin_loan, fin_card).
  - **Demographic Data:** Publicly available information about districts, potentially providing insights into client demographics (relation: fin_district).



**Client Information**

**Account Details**

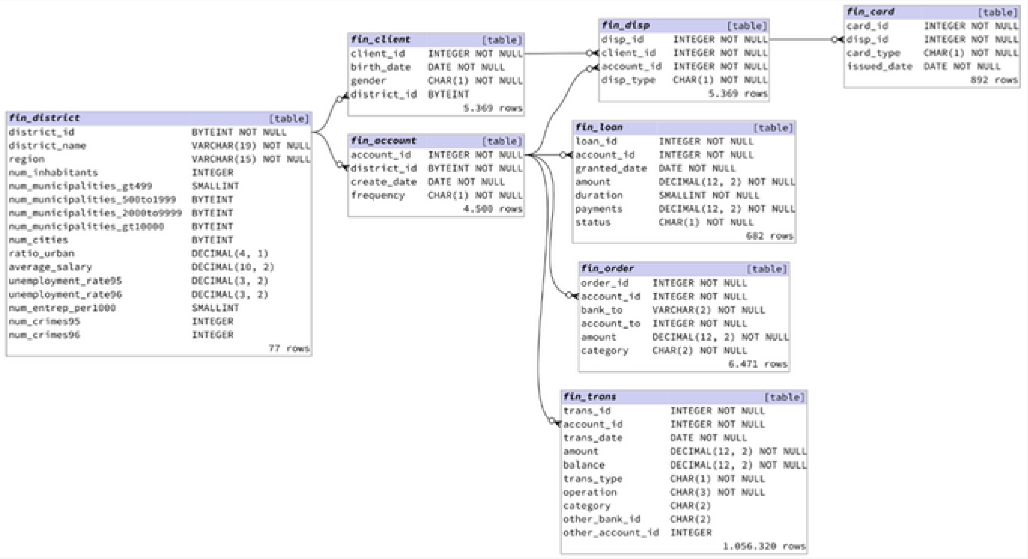
**Client-Account Relationships**

**Transactions**

**Financial Products**

**Demographic Data**

*https://github.com/dnoeth/1999_Czech_financial_dataset_Teradata*



| fin_district | [table] |
|---|---|
| district_id | BYTEINT NOT NULL |
| district_name | VARCHAR(19) NOT NULL |
| region | VARCHAR(15) NOT NULL |
| num_inhabitants | INTEGER |
| num_municipalities_gt499 | SMALLINT |
| num_municipalities_500to1999 | BYTEINT |
| num_municipalities_2000to9999 | BYTEINT |
| num_municipalities_gt10000 | BYTEINT |
| num_cities | BYTEINT |
| ratio_urban | DECIMAL(4, 1) |
| average_salary | DECIMAL(10, 2) |
| unemployment_rate95 | DECIMAL(3, 2) |
| unemployment_rate96 | DECIMAL(3, 2) |
| num_entrep_per1000 | SMALLINT |
| num_crimes95 | INTEGER |
| num_crimes96 | INTEGER |
| | 77 rows |

| fin_client | [table] |
|---|---|
| client_id | INTEGER NOT NULL |
| birth_date | DATE NOT NULL |
| gender | CHAR(1) NOT NULL |
| district_id | BYTEINT |
| | 5.369 rows |

| fin_account | [table] |
|---|---|
| account_id | INTEGER NOT NULL |
| district_id | BYTEINT NOT NULL |
| create_date | DATE NOT NULL |
| frequency | CHAR(1) NOT NULL |
| | 4.500 rows |

| fin_disp | [table] |
|---|---|
| disp_id | INTEGER NOT NULL |
| client_id | INTEGER NOT NULL |
| account_id | INTEGER NOT NULL |
| disp_type | CHAR(1) NOT NULL |
| | 5.369 rows |

| fin_card | [table] |
|---|---|
| card_id | INTEGER NOT NULL |
| disp_id | INTEGER NOT NULL |
| card_type | CHAR(1) NOT NULL |
| issued_date | DATE NOT NULL |
| | 892 rows |

| fin_loan | [table] |
|---|---|
| loan_id | INTEGER NOT NULL |
| account_id | INTEGER NOT NULL |
| granted_date | DATE NOT NULL |
| amount | DECIMAL(12, 2) NOT NULL |
| duration | SMALLINT NOT NULL |
| payments | DECIMAL(12, 2) NOT NULL |
| status | CHAR(1) NOT NULL |
| | 682 rows |

| fin_order | [table] |
|---|---|
| order_id | INTEGER NOT NULL |
| account_id | INTEGER NOT NULL |
| bank_to | VARCHAR(2) NOT NULL |
| account_to | INTEGER NOT NULL |
| amount | DECIMAL(12, 2) NOT NULL |
| category | CHAR(2) NOT NULL |
| | 6.471 rows |

| fin_trans | [table] |
|---|---|
| trans_id | INTEGER NOT NULL |
| account_id | INTEGER NOT NULL |
| trans_date | DATE NOT NULL |
| amount | DECIMAL(12, 2) NOT NULL |
| balance | DECIMAL(12, 2) NOT NULL |
| trans_type | CHAR(1) NOT NULL |
| operation | CHAR(3) NOT NULL |
| category | CHAR(2) |
| other_bank_id | CHAR(2) |
| other_account_id | INTEGER |
| | 1.056.320 rows |

*Fig 1 : Entity Relation Diagram for the Dataset*

# Tech Stack

- The project utilizes a blend of technologies to build the anomaly detection system.
- **Python** serves as the backbone for data manipulation and analysis, leveraging libraries like **pandas** and **scikit-learn**.
- Data is stored securely in **AWS S3** and **RDS**, where S3 talks with the model directly and RDS stores the data in tabular form.
- **AWS Glue** streamlines data extraction and transformation.
- Machine learning tools play an important role, with **SageMaker** offering a platform for training and deploying custom models if needed. Anomaly detection logic is primarily implemented in Python, with results stored in S3.
- Additionally, **AWS CloudWatch** monitors key metrics for performance insights. Visualization tools like **Tableau** can be integrated for further data exploration.
- Security is a priority, with **AWS Key Management Service (KMS)** safeguarding data throughout the process.

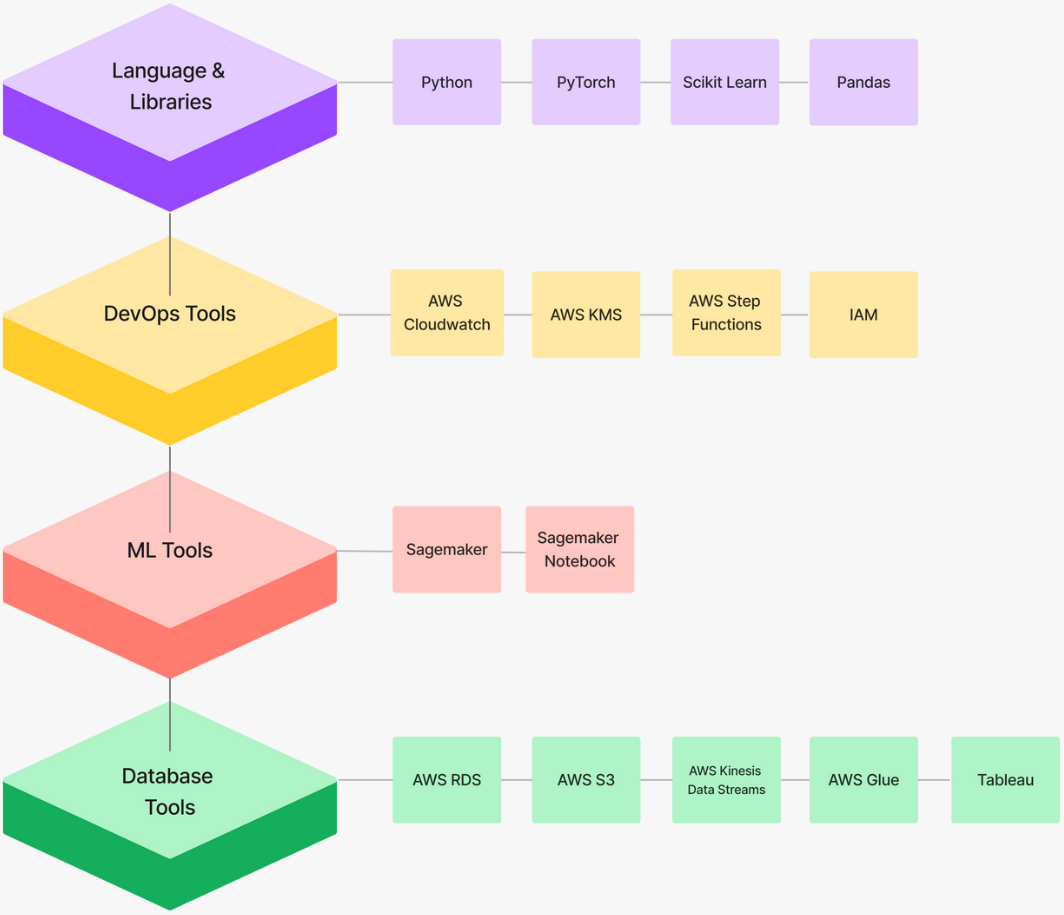


*Fig 2 : Tech Stack*

# Detailed System Design

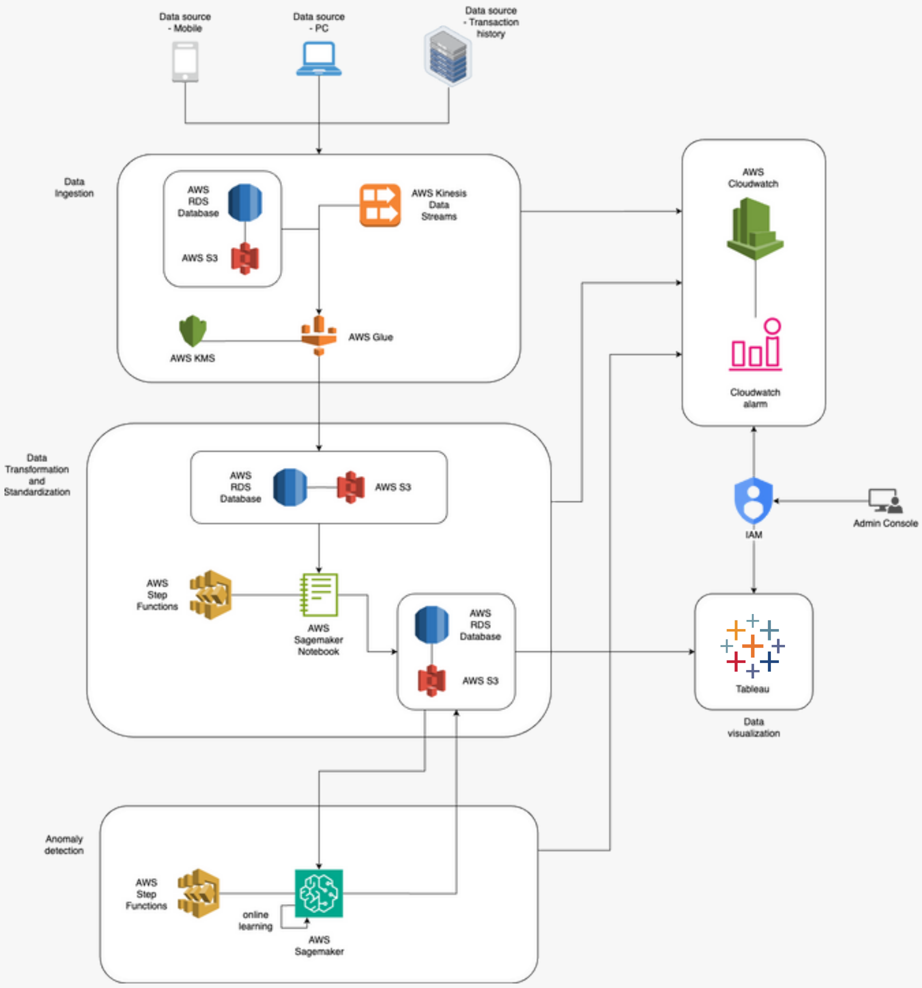


*Fig 3 : Architecture Diagram for the System*

Our anomaly detection system leverages a modular architecture, consisting of four distinct stages as illustrated in the accompanying diagram.

**1. Data Ingestion Module:**

- **Core Role:** Securely acquires financial transaction data, prepares it, and integrates it with a centralized data warehouse for future analysis.
- **Functionality:**
  - **Source Agnostic Design:** Handles data ingestion from various sources including real-time streaming data via platforms like AWS Kinesis and batch data from sources like AWS RDS.
- **Security:**
  - **Data Encryption:** Encrypts data at rest and in transit using industry-standard algorithms managed by AWS KMS.
  - **Network Security:** Ensures secure communication channels between data sources, ingestion pipelines, and processing systems using firewalls, IDS, and TLS protocols.

**2. Data Transformation & Standardization Module:**

- **Core Role:** Converts raw financial transaction data into a format suitable for anomaly detection models.
- **Functionality:**
  - **Data Cleaning & Pre-processing:** Addresses data quality issues like missing values, inconsistencies, and outliers using techniques like imputation, outlier removal, and normalization. Also performs data validation checks.
  - **Feature Engineering:** Creates informative features from existing data to improve anomaly detection.
- **Security:**
  - **Prevent Malicious Manipulation:** Implements data validation and sanitization to prevent attacks exploiting the transformation process.
  - **Code Security:** Utilizes secure coding practices and conducts regular audits to minimize vulnerabilities using SonarQube.

**3. Anomaly Detection Module:**

- **Core Role:** Uses machine learning algorithms to identify suspicious transactions in real or near real-time.
- **Functionality:**
  - **Automated Model Selection:** Utilizes AWS Step Functions to automate model selection through DAGs, allowing for a dynamic combination of models based on historical data.
  - **Adaptability:** Considers online learning to continuously update the model with legitimate transactions, adapting to changes like data drift or seasonality.
  - **Interpretability:** Integrates techniques like SHAP to provide human-interpretable insights into flagged anomalies, aiding the investigation.
- **Security:**
  - **Alerting Mechanisms:** Implements alerts for drops in model performance or unexpected data patterns.
  - **Model Security:** Securely stores trained models using encryption and access control to prevent unauthorized access or manipulation.

**4. Output & Visualization Module:**

- **Core Role:** Presents flagged anomalies and details for security investigation.
- **Functionality:**
  - **Real-time & Historical Presentation:** Displays real-time alerts and historical anomaly data for analysis.
  - **Interactive Visualizations:** Uses tools like Tableau to create interactive dashboards for exploring anomalies, including scatter plots, time series charts, and geographical heat maps.
- **Security:**
  - **Data Access Control:** Enforces granular access controls to restrict unauthorized access and implements data masking or anonymization for privacy.

# Implementation

The input dataset for anomaly detection module can have these features.

**Features:**

- Transaction ID
- Account ID
- Transaction Date
- Transaction Amount
- Transaction Type
- Client ID
- Client Age
- Client District ID
- Account Type
- Account Balance (Before)
- Account Balance (After)

**Client Details:**

- **client_id:** Links transactions to clients for identifying unusual spending patterns.
- **client_age:** Provides context for expected behaviour based on demographics.
- **client_district_id:** Incorporates insights from demographics (e.g., income levels).

**Account Details:**

- **account_type:** Helps identify anomalies specific to account types (savings vs. checking).
- **account_Balance(Before/After):** Enables calculating features like percentage change for anomaly detection.

**2. Local Outlier Factor (LOF):**

- **Concept:** Identifies anomalies by comparing a data point's local density deviation to neighbors.
- **Implementation:** Train LOF on enriched transaction data, assigning new transactions outlier factor scores. Higher scores imply higher outlier likelihood.

**3. LSTMs (Long Short-Term Memory):**

- **Concept:** RNN type learning sequential patterns, suitable for analyzing transaction time series.
- **Implementation:** Train LSTM on historical transaction sequences per client/account. New transactions predict subsequent behavior, with deviations signaling anomalies.

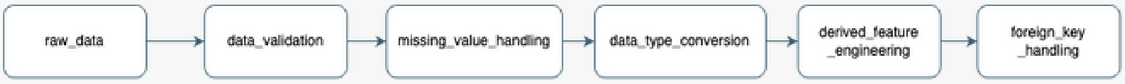


*Fig 4 : Data Transformation Pipeline*

We'll transfer the data to the anomaly detection module which uses Python libraries like PyTorch and Scikit Learn, and automate the process with AWS Step Functions. Model:

**1. Isolation Forest:**

- **Concept:** This algorithm isolates instances deviating significantly from the expected data pattern.
- **Implementation:** Train the model on the table. New transactions are scored based on isolation level, with higher scores indicating higher likelihood of anomalies.

**Detectable Anomalies:**

- ·Unusual spending sequences deviating from the client's typical transaction patterns.
- ·Transactions occurring outside the client's typical spending hours or locations
- ·Transactions exceeding the client's usual daily or monthly spending limits.
- ·Sudden bursts of transactions or transactions occurring in geographically unusual locations.

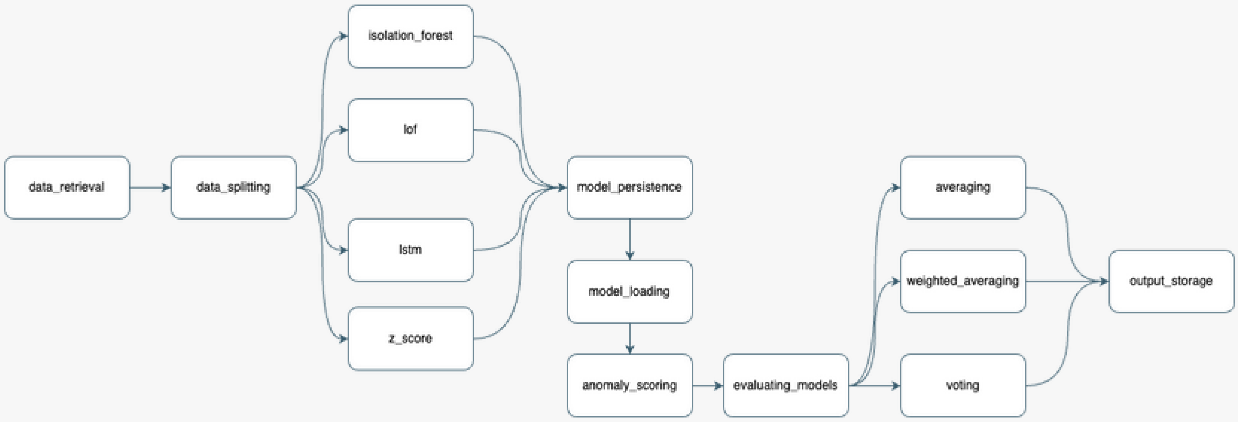


*Fig 5 : Anomaly Detection Pipeline*

**Additional features may include:**

- **Thresholding:** Define anomaly score thresholds for flagging transactions requiring further scrutiny.
- **Model Selection:** Optimal model choice depends on data characteristics and desired anomaly types. Experimentation with various models, including ensemble learning for improved detection, is essential.
- **Combining interpretability:** While Isolation Forest and LOF offer transparency, LSTMs can lack interpretability. Consider combining them with interpretable models for better anomaly understanding.
- **Robustness enhancement:** Implement online learning to adapt to evolving data patterns, including concept drift and seasonal trends. However, be aware of vulnerabilities to data tampering attacks, which can compromise the system's accuracy.

**Output:**

The output data can be stored in S3 from which tableau can retrieve data as it offers various options for connecting to data sources, one of them is:

- **Amazon Athena:** A serverless interactive query service that allows querying data directly in S3. Tableau can establish a live connection to Athena, enabling real-time visualization of the anomaly scores stored in the S3 bucket.

**Logging with CloudWatch:**

- **Data Ingestion:** Track volume (e.g., transactions/minute), errors (data source/network issues), and validation results (rejected records).

- **Transformation:** Log details (name changes, conversions), quality checks (missing values, outliers), and resource usage (CPU, memory).
- **Anomaly Detection:** Monitor metrics (precision, recall, F1 score), anomaly scores, and training details (data, hyperparameters).
- **Output:** Log delivery status (success/failure), visualization updates (Tableau), and alerts (specific anomalies, actions).

# And that's a close.

## Anomaly Detection System

- we have established a detailed plan and gathered initial requirements.
- we anticipate having completed a thorough analysis of potential data sources and designed the initial framework for our machine learning algorithms
- we are well-positioned with a refined and enhanced system, with a view towards a comprehensive solution for detecting and mitigating transactional anomalies

The Anomaly Detection System Framework we have developed addresses the critical need for accurate, reliable financial transaction processing amidst the challenges posed by operational anomalies and the rising threat of payment frauds, erroneous calculations, and missing prices. By leveraging advanced detection technologies, this framework ensures the integrity of transactions, enhances security measures, and boosts operational efficiency. As we move forward, this initiative stands as a testament to our commitment to safeguarding financial transactions and fostering trust within the financial ecosystem.