Integrating Data Validation with Large Language Models for Regulation-Guided Tabular Anomaly Detection

Haoliang Huang[1,*], Zihuang Cai[1,*], Zhuo Tang[1,†], Yifan Liu[1], Chen Tian[2], Kenli Li[1], Changjian Chen[1,†]
[1]College of Computer Science and Electronic Engineering, Hunan University,
[2]School of Computer Science, Nanjing University
{huanghl,zihuangcai,ztang,changjianchen}@hnu.edu.cn

Abstract

In many real-world applications, such as medical insurance, many regulations exist that define how data should comply with certain standards. Auditors typically use these regulations to identify anomalies in tabular data. However, existing tabular anomaly detection methods often focus on detecting anomalies based on data distribution without considering regulatory compliance. In this paper, we introduce a new task, **Regulation-guided Tabular Anomaly Detection**, which leverages regulations to detect anomalies in tabular data. We also developed three new datasets for this task. To address this task, we present **RegValidator**, a training-free method that integrates data validation with large language models (LLMs) for detecting anomalies. In this process, the LLMs generate ideas for anomaly detection from a regulation perspective, while the data validation validates these ideas from a data distribution perspective. This process can be framed as a Budgeted Maximum Coverage problem, which can be solved by a constant-factor approximation algorithm with provable guarantees. Empirical results on the new datasets demonstrate that our method outperforms existing baselines. A field experiment in a commercial health insurance company also reveals the practical value of our method. Our code is available at https://github.com/hnu-vis/RegValidator.

1 Introduction

In the era of big data, tabular data is one of the most prevalent data types (Jiang et al., 2025). The sheer volume of this data is accompanied by numerous anomalies, which are frequently associated with illicit activities (Hilal et al., 2022). To suppress and detect these anomalies, many regulations are introduced to define how data should comply with certain standards. With these regulations, auditors manually review the data to check compliance and detect the anomaly ones. However, when the data volume increases, this process becomes labor-intensive and time-consuming.

To tackle the issue of low efficiency, many automatic tabular anomaly detection methods have been proposed, ranging from traditional machine learning methods (e.g., Isolation Forest (Liu et al., 2008) and KNN (Ramaswamy et al., 2000)) to more recent deep learning methods (e.g., DTE (Livernoche et al., 2023)) and LLM-based methods (e.g., AnoLLM (Tsai et al., 2025)). These methods typically model the data distribution first. Then, they assess whether each sample individually violates this distribution to detect anomalies. While effective, these methods can fall short when anomalies require the combination of multiple samples for detection. For instance, in the medical insurance data shown in Figure 1, each sample appears normal when analyzed individually. However, the two samples related to patient "P-3872" are anomalous because it is impossible for one person to visit hospitals on the same day in both New York and Beijing. The existing methods are hard to detect such anomalies. Nevertheless, by incorporating regulations, such anomalies can be detected. For instance, consider the regulation in Figure 1: "The patient receiving the service must be the insurance beneficiary." From this, we can derive the detection idea that a patient can't receive medical services in two distant places at the same time. Using this idea, we can easily detect the two anomalous samples associated with patient "P-3872."

Although it is intuitive that incorporating regulations can enhance the performance of tabular anomaly detection, how to effectively integrate regulations into the detection process remains unclear, presenting two major challenges. **Challenge 1: regulation conversion**. Regulations typically outline the principles that data should follow. However, it remains unclear how to convert these principles into actionable detection ideas. It is also unclear how to utilize these detection ideas to retrieve anomalies from data. Although LLMs can be directly used for regulation conversion, our initial attempts revealed that many useless or wrong ideas are generated. **Challenge 2: anomaly data validation**. It is impossible to ensure that all generated ideas are accurate. Therefore, validating the anomalies in the samples retrieved by the ideas is crucial. However, existing methods are generally designed to detect individual anomalies, making it unclear how to validate anomalies in combinations of multiple samples.

To address these challenges, in this paper, we first introduce a new task, **R**egulation-guided **T**abular

[*]Both authors contributed equally to this work.
[†]Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6559–6581

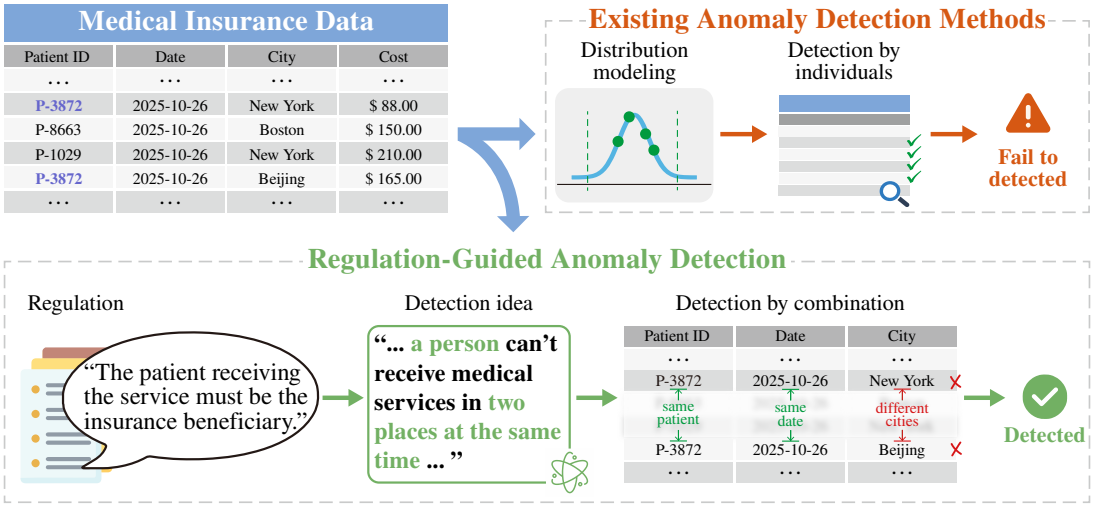

Figure 1: Comparison between the existing anomaly detection methods and our RegValidator.

Anomaly **D**etection (**RTAD**). We also construct three new datasets to evaluate the performance on this task. To address this task, we present **RegValidator**, a training-free method that integrates data validation with large language models for detecting anomalies. It consists of two main modules: a **multi-agent system** for regulation compliance and a **data validation method** for data distribution analysis. The multi-agent system extracts detection ideas from regulations and converts the ideas to SQL queries for retrieving anomalies. The data validation method generates several rules (e.g., unique_ratio is less than a threshold) to validate the retrieved anomalies. This validation process can be formulated as a Budgeted Maximum Coverage problem, which can be solved by a greedy algorithm (Khuller et al., 1999). We theoretically prove that our algorithm is $(\frac{1}{2} - \frac{1}{2e})$-approximation. We compare RegValidator against 16 baselines on the three datasets and find that our method outperforms the SOTA by 12.0% in AUC-ROC and 19.8% in F1-score. We also deploy RegValidator in a commercial health insurance company to demonstrate its practical values.

In summary, our contributions are threefold:

- A new paradigm, **regulation-guided anomaly detection**, and RegValidator, a training-free method that leverages regulations to enhance the performance of tabular anomaly detection.
- **An anomaly data validation method** with a theoretical guarantee of a $(\frac{1}{2} - \frac{1}{2e})$-approximation.
- **An empirical experiment** on three datasets and **a field experiment** in a commercial health insurance company demonstrate the effectiveness of the proposed method.

2 Related Work

LLMs for Anomaly Detection. Large Language Models (LLMs) have recently attracted significant attention in anomaly detection, demonstrating strong potential in analyzing complex data patterns where traditional statistical methods often struggle (Xu and Ding, 2025). Researchers have applied LLMs to various data modalities, exploring their capabilities in: capturing long-term temporal dependencies, using natural language prompts to identify segments that deviate from expected trends (Zhou and Yu, 2024; Liu et al., 2025; Park et al., 2025); detecting unusual visual elements or semantic inconsistencies in images and videos (Wu et al., 2024; Gu et al., 2024; Chen et al., 2024; Yang et al., 2024; Ye et al., 2025); and modeling sequential behaviors and detecting anomalies within system logs (Almodovar et al., 2024; Lim et al., 2025; Song et al., 2025). However, these methods are not tailored to tabular anomaly detection.

LLMs for Tabular Anomaly Detection. To bridge the gap between semantic LLMs and structured tables, many LLM-based tabular anomaly detection methods have been proposed. A common method is to linearize the samples in the tables into a natural language sentence, enabling an LLM to make an anomaly judgment (Li et al., 2024). AnoLLM follows this paradigm by fine-tuning LLMs to model the distribution of normal data, and using negative log-likelihood as an anomaly score (Tsai et al., 2025). Although effective at detecting samples that statistically deviate from the majority, these methods often struggle to detect violations of specific domain regulations. Because an individual sample may appear normal in isolation and remain statistically consistent with the overall distribution, yet violate the regulation when evaluated with related samples.

To incorporate domain logic, ReTabAD (Yoon et al., 2025) proposes a zero-shot LLM-based framework that injects domain knowledge, feature descriptions, and statistical summaries into prompts for context-aware anomaly detection. However, its reliance on the LLM context window limits anomaly detection to small sample batches and prevents reasoning over the full dataset or enforcing global constraints. As a result, ReTabAD is less effective for regulation-based anomaly detection, where violations often emerge through relationships

across numerous samples. In contrast, RegValidator applies explicit rules through database queries, enabling it to detect anomalies that violate regulations or involve relationships, which can scale to large-scale datasets.

3 Dataset Construction

Through our survey, we found that all the publicly available tabular anomaly detection datasets did not include the associated regulations. To fill this gap, inspired by the construction pipeline of (Lin et al., 2025), we construct three new datasets for the RTAD task: RTAD-Healthcare, RTAD-FDTD, and RTAD-Creditcard.

Dataset	Samples	Anomaly Ratio
Healthcare	558,211	38.1%
FDTD	50,000	32.0%
Credit Card	1,296,675	0.6%

Table 1: Statistics of the datasets.

Data Source. To ensure a comprehensive and robust evaluation, the three datasets vary in domain, size, and anomaly ratio, providing a diverse basis for dataset construction. Key statistics are summarized in Table 1.

- **Healthcare Provider Fraud Detection (Healthcare)** contains three tables for inpatient, outpatient, and beneficiary records, and a training table with provider-level labels.
- **Fraud Detection Transactions Dataset (FDTD)** contains simulated user transaction data with a relatively high proportion of anomalies.
- **Credit Card Transactions Fraud Detection (Credit Card)** is a large-scale dataset of simulated credit card transactions, featuring a highly imbalanced class distribution.

Data Pre-processing. For the Healthcare dataset, we use the provided training set for evaluation as the test set lacks anomaly labels. Since this dataset only identifies anomalous healthcare providers, we treated all records from these providers as anomalous. For the Credit Card Dataset, we only use the training set, which contains more samples than the test set.

3.1 Regulations

For each dataset, it is expected to include related regulations that can cover all the anomalies. All the regulations should be from real-world applications rather than being generated by LLMs. Therefore, we first retrieved regulations from Google using keywords (e.g., "healthcare" and "medical" for the Healthcare dataset). Then, we manually checked the retrieved regulations to ensure they were related to our datasets and covered all the anomalies. If they did not cover all the anomalies, we Google with more related keywords for more regulations. This process was repeated until the regulations covered all the anomalies, resulting the following ones. For the Healthcare dataset, we use *Medicare Fraud & Abuse: Prevent, Detect, Report*; *Fiscal Year 2023 Medicare and Medicaid Integrity Programs Report to Congress*; and *The False Claims Act: A Primer*. For the FDTD and Credit Card datasets, we use *The FATF Recommendations*; *FinCEN Suspicious Activity Report (FinCEN SAR) Electronic Filing Requirements*; *Indicators of suspicious activity for the banking sector*; and *Payment Card Industry Data Security Standard: Requirements and Testing Procedures, v4.0.1*.

3.2 Table Information

To leverage regulations for tabular anomaly detection, it is important to know the meanings of the tables and their columns. However, some of the datasets do not contain such information. For example, the Healthcare dataset does not contain the meaning of each column. Thus, we manually wrote the descriptions for the tables and their columns. For the table description, we recorded its background, size, and primary key. For each column, we provided a detailed description, data type, and sample values. If a column is numerical, we also included the maximum and minimum values for reference. In this work, we use LabelInspect for data processing (Liu et al., 2019).

4 Method

As shown in Figure 2, RegValidator takes regulations and the original tabular data stored in databases as input, and outputs the detected anomalous samples. It consists of two main modules: (1) **a multi-agent system** that extracts detection ideas from regulations and converts the ideas to SQL for retrieving anomalies; and (2) **a data validation method** that generates several rules to validate the retrieved anomalies. Since the data validation method is nested in the multi-agent system and its core is the validation rule generation, we first introduce the multi-agent system, then explain the rule generation and its theoretical guarantee.

4.1 Multi-Agent System

The multi-agent system consists of four agents: (1) an ideation agent to extract detection ideas from regulations; (2) a query agent to convert the detection ideas to SQL queries for retrieving anomaly candidates; (3) a rule generation agent to generate several rules for validation; (4) a validation agent to utilize the rules to compare the retrieved anomaly candidates and the entire data. All the steps processed by LLMs are represented by $f_{\cdot}^{\mathrm{LLM}}$, whose prompts are provided in the appendix I.

Ideation Agent. Regulations are lengthy, unstructured texts with detection ideas embedded in them. To extract these ideas, the ideation agent leverages the logical reasoning power of LLMs. The extraction process first identifies potential anomaly points from the data information, then retrieves relevant regulations to formulate detection ideas, and finally associates each detection idea with the database schema. This agent takes regulations (r) and data information (M) as inputs, and

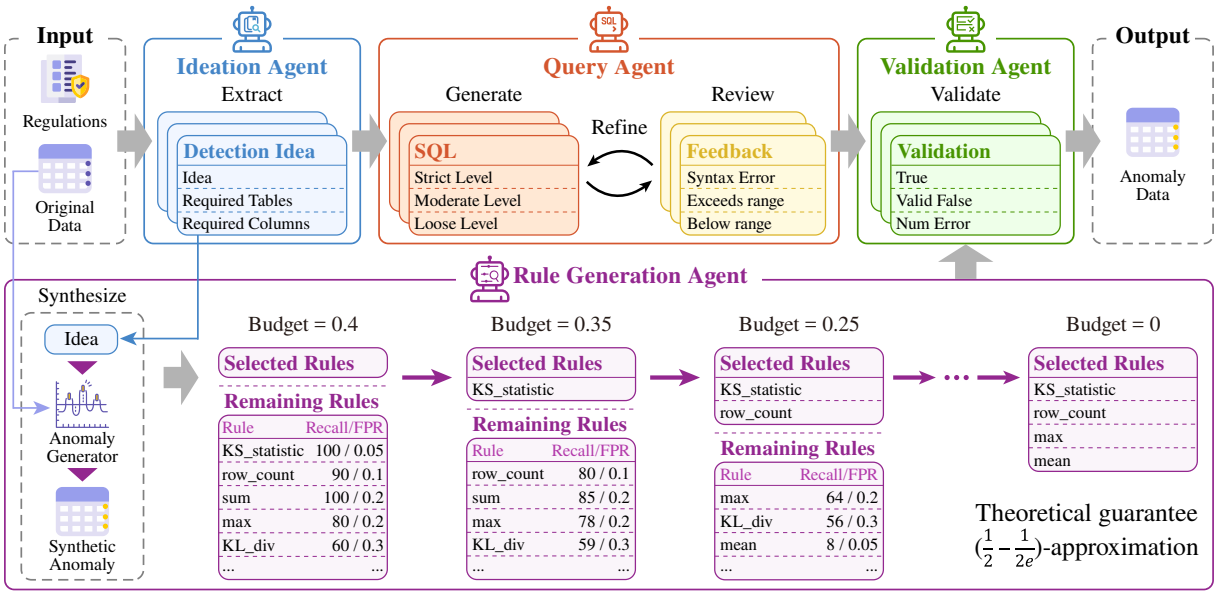

Figure 2: An overview of the RegValidator framework.

extracts a set of potential detection ideas (**I**):

$$\mathbf{I} = f^{\text{LLM}}_{\text{extract}}(r, M).$$

Here, $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$, and n denotes the total number of detection ideas. Each detection idea includes (1) a concrete description of the idea and (2) the required database tables and columns, which ensures compatibility with the database schema and supports accurate SQL generation in the following agents.

Query Agent. Given the detection ideas extracted by the ideation agent, the query agent converts each idea into executable SQL queries over the target dataset. Since directly generating correct queries is often challenging, this agent incorporates an iterative self-correction mechanism to generate and refine the queries. This iterative process can be expressed as:

$$\mathcal{Q}_i = f^{\text{LLM}}_{\text{query}}(I, M, \mathcal{Q}_{i-1}, \mathcal{H}_{i-1}), \quad (1)$$

$$\mathcal{H}_i = f_{\text{review}}(\mathcal{Q}_i). \quad (2)$$

Here, $\mathcal{Q}_i = \{q^{(s)}, q^{(m)}, q^{(l)}\}$ denotes the set of three SQL queries for the idea I at iteration i with varying strictness levels (strict(s), moderate(m), and loose(l)). $\mathcal{H}_i = \{h^{(s)}, h^{(m)}, h^{(l)}\}$ is the corresponding set of self-review feedback, where each $h^{\cdot}$ includes the syntax errors by executing the corresponding SQL query and an indicator of whether the query satisfies the current strictness level.

The process iterates as follows. Initially, $\mathcal{Q}_0$ is generated purely based on I and M ($\mathcal{Q}_0 = f^{\text{LLM}}_{\text{query}}(I, M)$), and $\mathcal{H}_0 = \emptyset$. In each iteration, the agent alternately performs query review (Eq. (2)) and query refinement (Eq. (1)). The review step executes the SQL query to collect syntax errors, and evaluate whether the scale of its returned results satisfies the strictness level. The scales of each strictness level are defined as ranges of the total number of samples: strict [1%,40%], moderate [2%,60%], loose [3%,80%]. Based on the review feedback from the previous iteration ($\mathcal{Q}_{i-1}, \mathcal{H}_{i-1}$), the refinement step modifies the current query by correcting syntax errors and adjusting thresholds, thereby improving both executability and effectiveness.

It is important to note that this iteration is not guaranteed to converge. Therefore, we cap the number of iterations at five. When the iteration terminates, we discard SQL queries that still produce syntax errors, while retaining those that are syntactically valid even if they do not satisfy the strictness level.

Rule Generation Agent. To determine whether the extracted detection ideas correspond to true anomalies, the rule generation agent associates each detection idea with a group of validation rules to validate its detection results. For each detection idea I, given data information (M), and original dataset ($D_{original}$), the validation rule V is obtained as:

$$V = f^{\text{LLM}}_{\text{rule}}(I, M, D_{original}).$$

After processing all detection ideas, we obtain the validation rules set $\mathbf{V} = \{V_1, V_2, \ldots, V_n\}$, where V_k is the validation rule for detection idea I_k. The rule generation method is described in 4.2.

Validation Agent. The validation agent assesses whether the generated SQL queries can effectively detect anomalies under the guidance of validation rules. This agent utilizes a soft validation strategy. Specifically, for each detection idea $I_k \in \mathbf{I}$, with its corresponding validation rule $V_k \in \mathbf{V}$, final query set $\mathcal{Q}^{(k)}$, and final feedback set $\mathcal{H}^{(k)}$, the agent assigns a validation weight to each query according to its validation results. For each query $q^{(\ell)}_k \in \mathcal{Q}^{(k)}$ with strictness level

Type	Metrics
Single-dataset	*min*, *max*, *mean*, *median*, *range* *row_count*, *unique_ratio*
Double-dataset	*JS_div*, *Cohen_dist*, *L-inf*, *KL_div* *KS_statistic*

Table 2: Statistical metrics used for rules. Detailed definitions are provided in Appendix B.

$\ell \in \{s, m, l\}$, the validation weight $\alpha_k^{(\ell)}$ is defined as:

$$\alpha_k^{(\ell)} = \begin{cases} 0.1s^{(\ell)}, & q_k^{(\ell)} \text{ violates the strictness level;} \\ 1.0s^{(\ell)}, & q_k^{(\ell)} \text{ passes the validation;} \\ 0.5s^{(\ell)}, & \text{otherwise.} \end{cases}$$

$s^{(\ell)}$ is applied because we found that SQL queries with stricter levels tend to be more accurate. Therefore, we assign higher weights to such queries. Accordingly, $s^{(\ell)}$ is set to 0.5, 0.3, and 0.2 for the strict, moderate, and loose levels, respectively.

With the validation weights for the SQL queries, we can obtain the anomaly score of each sample as follows. For a sample x, its anomaly score is computed by aggregating its weighted contributions across all SQL queries that retrieve it.

$$\text{Score}(x) = \sum_{k=1}^{n} \sum_{\ell \in \{s,m,l\}} \chi(x, q_k^{(\ell)}) \cdot \alpha_k^{(\ell)},$$

where $\chi(x, q)$ is an indicator function that equals 1 if sample x is retrieved by query q, and 0 otherwise.

4.2 Validation Rule Generation

The goal of the validation rules is to identify SQL queries that detect as many anomalies as possible (high recall) while introducing as few false positives as possible (low false-positive rate (FPR)). However, as our method is training-free, there are no anomaly labels for evaluating recall and FPR scores. Following the work of AVH (Tu et al., 2023), we utilize a generation-and-evaluation strategy, where we generate several synthetic anomalies and evaluate our rules on them. The problem formulation and analysis framework are inspired by (Xiang et al., 2025).

Problem Formulation. We consider a rule space $\mathbf{R}$, where each rule $R \in \mathbf{R}$ computes a statistic ϕ on a set of columns C over the target dataset D and compares it with given thresholds. There are two types of rules: single-dataset and double-dataset rules. The single-dataset rule $R(D, C, \theta_{\text{low}}, \theta_{\text{up}})$ can be expressed as $\phi_{\text{single}}(D, C) \notin [\theta_{\text{low}}, \theta_{\text{up}}]$. Similar to AVH (Tu et al., 2023), ϕ_{single} is calculated on a single dataset. The double-dataset rule $R(D, D', C, \theta_{\text{low}}, \theta_{\text{up}})$ can be expressed as $\phi_{\text{double}}(D, D', C) \notin [\theta_{\text{low}}, \theta_{\text{up}}]$, where D' is a reference dataset, usually represented by several randomly selected samples from the original data. ϕ_{double} is calculated on both D and D'. Table 2 presents the statistical metrics used in our experiments. Other rules can also be included.

Since our goal is to improve the recall while maintaining a minimum FPR. It can be formulated by selecting a subset of rules V that maximizes the recall while ensuring the FPR does not exceed a given budget:

$$\begin{aligned} \max \quad & \text{Recall}(P(V)) \\ \text{s.t.} \quad & \text{FPR}(P(V)) \le \delta \\ & P(V) = \bigcup_{R_i \in V} S(R_i). \end{aligned} \tag{3}$$

Here, $S(R_i)$ denotes the set of samples detected as anomalous by rule R_i. $P(V)$ corresponds to the union of these detected samples. $\text{Recall}(\cdot)$ and $\text{FPR}(\cdot)$ are the recall and FPR of $P(V)$, respectively. δ is the FPR budget.

Solving Eq. (3) requires anomaly labels to calculate recall and FPR, which are not available in our setting. We therefore discuss how to estimate them.

FPR estimation. The FPR of a rule R is the proportion of normal samples incorrectly detected as anomalous by R. According to Chebyshev's inequality, we can obtain the error bound of the expected FPR even without the anomaly labels. Specifically, we randomly sample t subsets of size m from the original dataset D_{original}. We denote these subsets as $D_{\text{sample}} = \{D_1, \ldots, D_t\}$, which reflect the main distribution of the original dataset, samples of them that satisfy R can be treated as false positives. For each statistic ϕ, we compute its values on the sampled datasets as $\phi(D_{\text{sample}}, C) = (\phi(D_1, C), \ldots, \phi(D_t, C))$, from which we estimate the sample mean μ and variance σ^2. Given a tolerance parameter $\beta > 0$, the thresholds are set as $\theta_{\text{low}} = \mu - \beta$ and $\theta_{\text{up}} = \mu + \beta$. By Chebyshev's inequality, the expected FPR is bounded by:

$$\text{E}[\text{FPR}(R)] \le (\sigma/\beta)^2. \tag{4}$$

The correctness of Eq. (4) is proved in the Appendix C. Since Eq. (3) is upper bounded by an FPR budget, using the bound in Eq. (4) will not violate the budget constraint. Therefore, we directly Eq. (4) as the FPR estimate: $\text{FPR}(R) = (\sigma/\beta)^2$.

Recall estimation. To evaluate the recall, we generate synthetic anomaly samples based on the detection ideas. Given a detection idea $I \in \mathbf{I}$, the table information M, we first utilize an LLM to identify the most related column $C_{\text{key}} = f_{\text{select}}^{\text{LLM}}(I, M)$. Then, we sample a subset of the data of size m and apply random perturbations on their C_{key} column to obtain the synthetic anomaly samples D_{syn}. With D_{syn}, the recall of a rule R can be estimated by $\text{Recall}(R) = |\text{SatiSet}(R)|/m$. Here, $\text{SatiSet}(R)$ is defined as:

$$\text{SatiSet}(R) = \{x \mid x \in D_{\text{syn}}, x \text{ satisfies } R\}. \tag{5}$$

Then, for a set V of multiple rules, its recall can be estimated by $\text{Recall}(P(V)) = \left|\bigcup_{R_i \in V} \text{SatiSet}(R_i)\right|/m$,

Algorithm 1 Rule Selection under FPR Budget

Input: Candidate set $\mathbf{R}$, budget δ
Output: Selected validation rule set V

$V \leftarrow \emptyset$, $U \leftarrow \mathbf{R}$
while $U \neq \emptyset$ **do**
  $R_s \leftarrow \arg\max_{R_i \in U} W_i / c_i$
  if $\sum_{R_i \in V} c_i + c_s \leq \delta$ **then**
    $V \leftarrow V \cup \{R_s\}$
  end if
  $U \leftarrow U \setminus \{R_s\}$
end while
$R_t \leftarrow \arg\max_{R \in \mathbf{R}} w(\{R\})$
if $w(V) \geq w(\{R_t\})$ **then**
  return V
else
  return $\{R_t\}$
end if

and its FPR can be estimated by $\sum_{R_i \in V} \mathrm{FPR}(R_i)$. Here, $P(V) = \bigcup_{R_i \in V} R_i$

With the recall and FPR estimation, the Eq. (3) can be reformulated into a *Budgeted Maximum Coverage* problem:

$$\begin{aligned} &\max \left| \bigcup_{R_i \in V} \mathrm{SatiSet}(R_i) \right| \\ &\text{s.t.} \sum_{R_i \in V} \mathrm{FPR}(R_i) \leq \delta. \end{aligned} \tag{6}$$

Approximation Algorithm. The Budgeted Maximum Coverage problem is NP-hard, making exact optimization infeasible for large rule spaces. To efficiently construct a near-optimal solution, following the method in (Khuller et al., 1999), we adopt a greedy approximation algorithm.

This algorithm maintains a selected set V and a remaining set U. Initially, $V = \emptyset$ and $U = \mathbf{R}$. At each iteration, the algorithm selects the rule R_i that maximizes the ratios of marginal coverage gain to cost, W_i / c_i. Here, the marginal coverage gain $W_i = |\mathrm{SatiSet}(R_i) \setminus \bigcup_{R_j \in V} \mathrm{SatiSet}(R_j)|$. The cost $c_i = \mathrm{FPR}(R_i)$. If adding R_i to V does not exceed the budget (i.e., $\sum_{R_j \in V} c_j + c_i \leq \delta$), it is included in V; otherwise, it is discarded. In both cases, R_i is removed from U, ensuring each candidate is considered at most once. This process iterates until $U = \emptyset$.

After the greedy iteration, the algorithm also selects a single-rule solution $\{R_t\}$ with the largest coverage $w(\{R_t\}) = |\mathrm{SatiSet}(R_t)|$. If the coverage of $\{R_t\}$ ($w(\{R_t\})$) is larger than that of V ($w(V)$), the algorithm returns $\{R_t\}$; otherwise, it returns V. Here, $w(V) = |\bigcup_{R_i \in V} \mathrm{SatiSet}(R_i)|$. This step is crucial for guaranteeing a provable approximation factor for the algorithm. The detailed process is summarized in Algorithm 1.

Theoretical Analysis. We can prove that the greedy approximation algorithm is $(\frac{1}{2} - \frac{1}{2e})$-approximation.

Theorem 1 *Let $OPT \subseteq \mathbf{R}$ denote the optimal rule set satisfying the budget constraint δ. Let V_{greedy} denote the output of Algorithm 1. Then, the total coverage achieved by V_{greedy} satisfies*

$$w(V_{greedy}) \geq \left(\frac{1}{2} - \frac{1}{2e}\right) w(OPT),$$

where $w(\cdot)$ denotes the anomalies covered by rules.

The proof of Theorem 1 is provided in Appendix D. This theorem guarantees that our greedy approximation algorithm can always obtain a reasonably good solution within a feasible time.

5 Experiments

In this section, we introduce a series of experiments to evaluate the performance of RegValidator.

5.1 Experimental Settings

Baselines. We compared RegValidator against three categories of representative baselines: (1) *Classical baselines*: we evaluated four representative classical anomaly detection methods, including Isolation Forest (Liu et al., 2008), PCA (Shyu et al., 2003), KNN (Ramaswamy et al., 2000), and ECOD (Li et al., 2022). (2) *Deep learning–based baselines*: for this category, we considered DeepSVDD (Ruff et al., 2018), RCA (Liu et al., 2021), SLAD (Xu et al., 2023b), GOAD (Bergman and Hoshen, 2020), NeuTral (Qiu et al., 2021), ICL (Shenkar and Wolf, 2022), DTE (Livernoche et al., 2023), REPEN (Pang et al., 2018), RDP (Wang et al., 2019), and DIF (Xu et al., 2023a). (3) *LLM–based baselines*: we included ReTabAD and AnoLLM (Tsai et al., 2025), the state-of-the-art method that utilizes LLMs to detect anomalies in tabular data.

Implementation Details. For ReTabAD and RegValidator, all agents were powered by the GPT-4o-mini. To balance the effectiveness and computational efficiency of FPR estimation in the rule generation agent, we randomly sampled 50 subsets from the original dataset, with 1,000 samples per subset. The FPR threshold δ is set to 0.4. For all the baselines, we followed their official implementations and hyperparameter settings to ensure a fair comparison. For AnoLLM, we adopt SmolLM-135M, with fine-tuning and inference conducted on an NVIDIA A100 40GB GPU.

Evaluation Metrics. We evaluated all methods using two standard metrics: **AUC-ROC** and **F1-score**. All the methods were evaluated on the three datasets presented in Sec. 3. For AnoLLM and non-LLM baselines, an unsupervised setting was used. Half of the normal samples are used for training, and the remaining normal samples, together with all anomalous samples are used for testing. For ReTabAD and RegValidator, a training-free setting was used. The same set is used for testing, but no samples are used for training. For the ReTabAD baseline, it is impossible to input the entire data into LLMs, especially the RTAD-Creditcard dataset with more than

Method	RTAD-Healthcare		RTAD-FDTD		RTAD-Creditcard		Average	
	AUC-ROC	F1	AUC-ROC	F1	AUC-ROC	F1	AUC-ROC	F1
Classical baselines								
Isolation Forest	0.515	0.563	0.564	0.535	0.616	0.025	0.565	0.374
PCA	0.514	0.565	0.703	0.652	0.612	0.024	0.610	0.414
KNN	0.542	0.577	0.719	0.655	0.834	0.299	0.699	0.510
ECOD	0.511	0.559	0.605	0.563	0.644	0.026	0.586	0.383
Deep Learning-based baselines								
DeepSVDD	0.507	0.556	0.466	0.462	0.680	0.117	0.551	0.378
RCA	0.527	0.567	0.794	0.732	0.833	0.334	0.718	0.545
SLAD	0.519	0.562	0.509	0.492	0.506	0.012	0.511	0.355
GOAD	0.517	0.561	0.831	0.739	0.660	0.039	0.670	0.446
NeuTral	0.514	0.560	0.634	0.587	0.649	0.022	0.599	0.389
ICL	0.527	0.570	0.814	0.738	0.742	0.030	0.694	0.445
DTE	0.497	0.550	0.502	0.487	0.500	0.012	0.500	0.350
REPEN	0.522	0.564	0.527	0.511	0.769	0.088	0.606	0.388
RDP	0.523	0.565	0.633	0.587	0.733	0.157	0.629	0.436
DIF	0.527	0.566	0.585	0.552	0.791	0.137	0.634	0.418
LLM-based baselines								
ReTabAD	0.489	0.590	0.601	0.595	0.766	0.184	0.618	0.456
AnoLLM	0.544	0.576	0.870	0.793	0.649	0.033	0.688	0.467
RegValidator	**0.638**	**0.710**	**0.945**	**0.879**	**0.842**	**0.405**	**0.808**	**0.665**

Table 3: Main results comparing RegValidator with baseline methods across three datasets. The best score in each column is shown in **bold**, and the second-best is underlined.

Method	RTAD-Healthcare		RTAD-FDTD		RTAD-Creditcard		Average	
	AUC-ROC	F1	AUC-ROC	F1	AUC-ROC	F1	AUC-ROC	F1
RegValidator	**0.638**	**0.710**	**0.945**	**0.879**	**0.842**	**0.405**	**0.808**	**0.665**
w/o data validation	0.637	**0.710**	0.865	0.784	0.824	0.339	0.775	0.611

Table 4: Ablation study results across all three datasets. The best score in each column is shown in **bold**.

one million samples. Therefore, we randomly sample 1,000 samples for testing, while maintaining the same proportion of anomalous samples as in the full dataset. To evaluate the methods, we consider the top-K samples with the highest anomaly scores as anomalous, where K corresponds to the true number of anomalies in each dataset. All reported results are averaged over three independent runs.

5.2 Main Results

Table 3 presents the main experimental results across the three datasets. The results demonstrate that RegValidator achieves state-of-the-art performance, consistently outperforming all baseline methods across all datasets. For example, on the RTAD-Creditcard dataset, all the baselines perform poorly, with most of them achieving F1-scores below 0.2. In contrast, our method achieves an F1-score of 0.405 on this dataset, substantially improving performance. Compared to AnoLLM, the SOTA tabular anomaly detection method, our method shows consistent improvement. Overall, our method achieves an average improvement of 12.0% in AUC-ROC and 19.8% in F1-score over AnoLLM. This result demonstrates that combining Regulation-based reasoning with data validation captures real-world anomaly patterns more effectively.

5.3 Ablation Study

To validate the contribution of the data validation method in RegValidator, we conducted ablation studies across all three datasets. We designed one variants: **RegValidator w/o data validation**, which disables the data validation method used to validate detection ideas.

The results are presented in Table 4. As we can see, our method consistently outperforms RegValidator w/o data validation across all three datasets. On average, our method achieves an improvement of 3.3% in AUC-ROC and 5.4% in F1-score.

5.4 Case Study

To better illustrate the effectiveness of RegValidator and the data validation method, we provide two case studies. More case studies can be found in the Appendix H.

Effectiveness of RegValidator. We first analyze the advantages of our method in detecting anomalies that

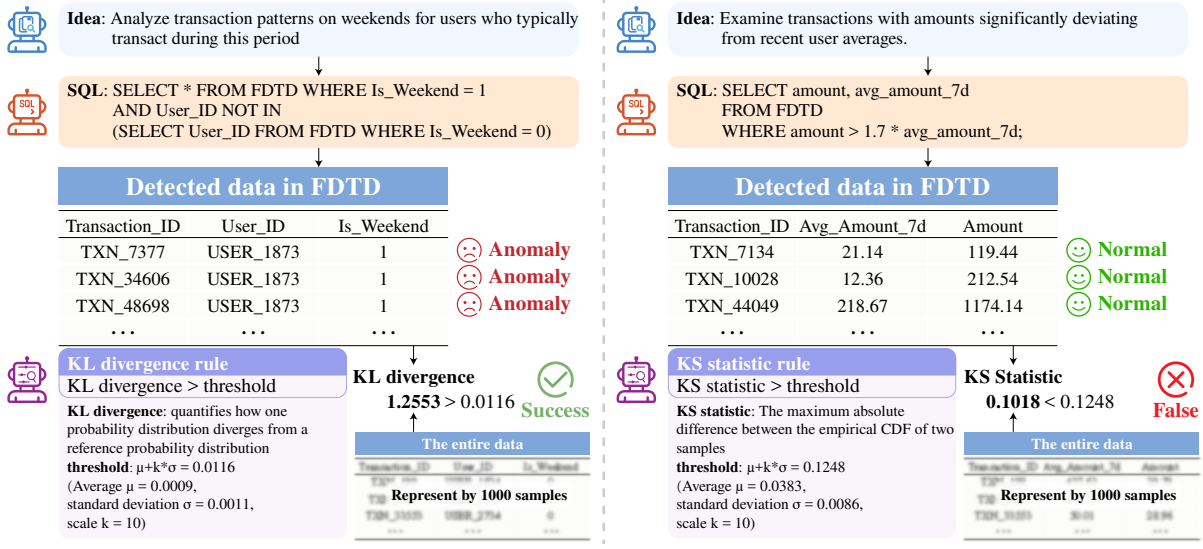

Figure 3: Two cases to show how the data validation method works in anomaly detection.

require combining multiple samples. Taking the RTAD-FDTD dataset as an example, the samples shown in Figure 3 are anomalies, as transactions occur **only on weekends** when few or no auditors are at work. All baselines failed to detect these anomalies from a data-distribution perspective. For these baselines, each sample appears normal when considered individually, since it is common to have transactions on weekends. However, from the regulation "Indicators of suspicious activity for the banking sector," our method extracted the detection idea "Analyze transaction patterns on weekends for users who typically transact during this period." With this idea, our method successfully detected these anomalous samples with the SQL generated from the detection idea. This case shows that our method can effectively combine multiple samples for anomaly analysis with the guidance of regulations.

Effectiveness of the Data Validation Method. We also provide a detailed case to demonstrate how the data validation method works. Figure 3 shows two examples in the RTAD-FDTD dataset. The left example in Figure 3 shows how the data validation method validated a correct detection idea. Based on the idea, our method first retrieves several anomalous candidates by generating an SQL query. Then, our data validation method generates a validation rule, the Kullback-Leibler (KL) divergence rule, that has a high recall but a low false positive rate. Then, this rule is applied to compare the retrieved samples and the entire data (represented by a set of randomly selected samples for computational efficiency). The KL divergence between them is large (1.2553) because these anomalous candidates occur only during the weekend, whereas normal users transact on both weekdays and weekends, which is larger than the threshold. Thus, this idea is validated as correct.

The right example Figure 3 shows how the data validation method validated a wrong detection idea. This idea states that transactions with amounts significantly deviating from recent user averages are anomalies. This is not always true because users may occasionally make expensive purchases, which leads to higher transaction amounts. Similar to the left example, a validation rule, Kolmogorov-Smirnov (KS) Statistic rule, is used. However, since this detection idea is wrong, the retrieved samples are not anomalies. Therefore, the KS statistic score is less than the threshold, meaning this idea does not pass the validation and thus successfully filtered out.

5.5 Field Experiment

In some real-world applications, generating detection ideas can be more valuable than merely detecting individual anomalies, as they can reveal systematic illicit activities. Our method naturally generates such detection ideas, thus can be used in such applications. To evaluate its effectiveness, we conducted a ten-month collaboration with a commercial health insurance company to analyze anomalies in its private data. The dataset includes 45 tables with a total of 4,529,777,353 samples, and the regulations involve 13 documents. Our method identifies 40 detection ideas from this data.

To evaluate the correctness of these identified ideas, we compared them with 28 detection ideas previously identified by the company. We found that 27 of the company's detection ideas were successfully identified. The remaining one was not identified because it is related to medical guidelines that were not mentioned in the regulations corpus. In addition, among the remaining 13 ideas identified by our method, the company confirmed that 12 of them pinpoint real illicit activities. The auditors of the company pointed out that they haven't been able to uncover new ideas for a long time. Our method can identify new ideas much more quickly, which not only improves efficiency but also increases recall.

6 Conclusion

In this work, we introduced Regulation-guided Tabular Anomaly Detection (RTAD) and proposed RegValidator, the first framework that leverages regulations to enhance anomaly detection in tabular data. RegValidator combines a multi-agent system, which extracts detection ideas from regulations and translates them into SQL queries, with a data validation method that verifies retrieved anomalies with a $(\frac{1}{2} - \frac{1}{2e})$-approximation guarantee. Experiments on three constructed datasets show that RegValidator outperforms state-of-the-art baselines. A field deployment in a commercial health insurance company confirmed its ability to detect both known and new anomalies. This work paves the way for applying regulation-guided anomaly detection in real-world scenarios, such as auditing and fraud prevention.

Limitations

We discuss the limitations of our work from two aspects. First, the datasets we constructed cover only two domains, healthcare insurance and financial transactions, which are relatively limited. Future work can expand to larger datasets across more domains. Second, the rules in our Rule Generation are predefined and based on common statistics. Going forward, we could explore ways to generate more effective and efficient rules that better validate the results.

Ethical Considerations

This research complies with the ethical guidelines of the Association for Computational Linguistics (ACL). The three datasets we constructed are based on publicly available tabular datasets from Kaggle and are used for academic purposes. They do not contain any personally identifiable information or sensitive user data, ensuring no direct privacy risks. Additionally, all regulations used in our study are publicly accessible.

For the field experiment conducted with the health insurance company, all data were anonymized and do not contain any personal or sensitive information. The large language model was deployed on local servers, ensuring that no data processed during the experiment was exposed or leaked. This experiment was approved by the company and aimed to detect potentially fraudulent activities effectively.

Furthermore, AI assistants were used to support the writing for polishing. All outputs generated by AI were carefully reviewed and verified by human authors.

In conclusion, we ensured comprehensive privacy protection and responsible use of AI throughout this research.

Acknowledgement

The work is supported by the National Natural Science Foundation of China (Grant Nos. 62225205, 62532005, 62402167), the Science and Technology Program of Changsha (kh2301011), the Major Science and Technology Research Projects of Hunan Province (Grant Nos. 2024QK2010, 2024QK2009), the Yunnan Provincial Major Science and Technology Special Plan Projects (Grant No. 202502AD080009), the Yunnan Science and Technology Talents and Platforms Program (202605AK340003), Project of Yuelushan Center for Industrial Innovation (Grant No. 2025TCII0206), Yuelushan Laboratory Breeding Program (Grant No. YLS-2025-ZY01015), the Hunan Natural Science Foundation (Grant No. 2025JJ60419), and the Science and Technology Innovation Program of Hunan Province (Grant No. 2023ZJ1080).

References

Crispin Almodovar, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad. 2024. LogFiT: Log anomaly detection using fine-tuned language models. *IEEE Transactions on Network and Service Management*, 21(2):1715–1723.

Liron Bergman and Yedid Hoshen. 2020. Classification-based anomaly detection for general data. *arXiv preprint arXiv:2005.02359*.

Cyrus D Cantrell. 2000. *Modern mathematical methods for physicists and engineers*. Cambridge University Press.

Xuhai Chen, Jiangning Zhang, Guanzhong Tian, Haoyang He, Wuhao Zhang, Yabiao Wang, Chengjie Wang, and Yong Liu. 2024. Clip-ad: A language-guided staged dual-path model for zero-shot anomaly detection. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 17–33.

Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. routledge.

Dominik Maria Endres and Johannes E Schindelin. 2003. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860.

Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, and Jinqiao Wang. 2024. Anomalygpt: Detecting industrial anomalies using large vision-language models. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1932–1940.

Waleed Hilal, S Andrew Gadsden, and John Yawney. 2022. Financial fraud: a review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429.

Jun-Peng Jiang, Si-Yang Liu, Hao-Run Cai, Qile Zhou, and Han-Jia Ye. 2025. Representation learning for tabular data: A comprehensive survey. *arXiv preprint arXiv:2504.16109*.

Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

Aodong Li, Yunhan Zhao, Chen Qiu, Marius Kloft, Padhraic Smyth, Maja Rudolph, and Stephan Mandt. 2024. Anomaly detection of tabular data using llms. *arXiv preprint arXiv:2406.16308*.

Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George H Chen. 2022. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12181–12193.

Ying Fu Lim, Jiawen Zhu, and Guansong Pang. 2025. Adapting large language models for parameter-efficient log anomaly detection. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 325–337.

Minzhi Lin, Tianchi Xie, Mengchen Liu, Yilin Ye, Changjian Chen, and Shixia Liu. 2025. InfoChartQA: A benchmark for multimodal question answering on infographic charts. In *In proceedings of the Neural Information Processing Systems (accepted)*.

Boyang Liu, Ding Wang, Kaixiang Lin, Pang-Ning Tan, and Jiayu Zhou. 2021. RCA: A deep collaborative autoencoder approach for anomaly detection. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1505–1511.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of IEEE International Conference on Data Mining*, pages 413–422.

Jun Liu, Chaoyun Zhang, Jiaxu Qian, Minghua Ma, Si Qin, Chetan Bansal, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2025. Large language models can deliver accurate and interpretable time series anomaly detection. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4623–4634.

Shixia Liu, Changjian Chen, Yafeng Lu, Fangxin Ouyang, and Bin Wang. 2019. An interactive method to improve crowdsourced annotations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):235–245.

Victor Livernoche, Vineet Jain, Yashar Hezaveh, and Siamak Ravanbakhsh. 2023. On diffusion modeling for anomaly detection. *arXiv preprint arXiv:2305.18593*.

Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. 2018. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2041–2050.

Min-Yeong Park, Won-Jeong Lee, Seong Tae Kim, and Gyeong-Moon Park. 2025. When will it fail?: Anomaly to prompt for forecasting future anomalies in time series. *arXiv preprint arXiv:2506.23596*.

Chen Qiu, Timo Pfrommer, Marius Kloft, Stephan Mandt, and Maja Rudolph. 2021. Neural transformation learning for deep anomaly detection beyond images. In *Proceedings of the International Conference on Machine Learning*, pages 8703–8714.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 427–438.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *Proceedings of the International Conference on Machine Learning*, pages 4393–4402.

Tom Shenkar and Lior Wolf. 2022. Anomaly detection for tabular data with internal contrastive learning. In *Proceedings of the International Conference on Learning Representations*.

Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. 2003. A novel anomaly detection scheme based on principal component classifier.

Marek Sikora and Lukasz Wrobel. 2010. seismic-bumps. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5W902.

Nikolai V Smirnov. 1939. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Bull. Math. Univ. Moscou*, 2(2):3–14.

Shuang Song, Yifei Zhang, and Neng Gao. 2025. Confront Insider Threat: Precise anomaly detection in behavior logs based on llm fine-tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8589–8601.

Che-Ping Tsai, Ganyu Teng, Phillip Wallis, and Wei Ding. 2025. Anollm: Large language models for tabular anomaly detection. In *Proceedings of the Thirteenth International Conference on Learning Representations*.

Dezhan Tu, Yeye He, Weiwei Cui, Song Ge, Haidong Zhang, Shi Han, Dongmei Zhang, and Surajit Chaudhuri. 2023. Auto-validate by-history: auto-program data quality constraints to validate recurring data pipelines. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4991–5003.

Hu Wang, Guansong Pang, Chunhua Shen, and Congbo Ma. 2019. Unsupervised representation learning by predicting random distances. *arXiv preprint arXiv:1912.12186*.

Peng Wu, Xuerong Zhou, Guansong Pang, Lingru Zhou, Qingsen Yan, Peng Wang, and Yanning Zhang. 2024. Vadclip: Adapting vision-language models for weakly supervised video anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6074–6082.

Ting Xiang, Changjian Chen, Zhuo Tang, Qifeng Zhang, Fei Lyu, Li Yang, Jiapeng Zhang, and Kenli Li. 2025. Enhancing small-scale dataset expansion with triplet-connection-based sample re-weighting. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10054–10063.

Hongzuo Xu, Guansong Pang, Yijie Wang, and Yongjun Wang. 2023a. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12591–12604.

Hongzuo Xu, Yijie Wang, Juhui Wei, Songlei Jian, Yizhou Li, and Ning Liu. 2023b. Fascinating supervisory signals and where to find them: Deep anomaly detection with scale learning. In *Proceedings of International Conference on Machine Learning*, pages 38655–38673.

Ruiyao Xu and Kaize Ding. 2025. Large language models for anomaly and out-of-distribution detection: A survey. In *Proceedings of Findings of the Association for Computational Linguistics: NAACL*, pages 5992–6012.

Yuchen Yang, Kwonjoon Lee, Behzad Dariush, Yinzhi Cao, and Shao-Yuan Lo. 2024. Follow the rules: Reasoning for video anomaly detection with large language models. In *Proceedings of European Conference on Computer Vision*, pages 304–322.

Muchao Ye, Weiyang Liu, and Pan He. 2025. Vera: Explainable video anomaly detection via verbalized learning of vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8679–8688.

Sanghyu Yoon, Dongmin Kim, Suhee Yoon, Ye Seul Sim, Seungdong Yoa, Hye-Seung Cho, Soonyoung Lee, Hankook Lee, and Woohyung Lim. 2025. ReTabAD: A benchmark for restoring semantic context in tabular anomaly detection. *arXiv preprint arXiv:2510.02060*.

Zihao Zhou and Rose Yu. 2024. Can llms understand time series anomalies? *arXiv preprint arXiv:2410.05440*.

A Dataset Details

We provide a comprehensive list of all regulations incorporated in the constructed datasets, along with their corresponding sources. All regulations are publicly available and summarized in Table 5.

Subsequently, we present the data information compiled for the three datasets. The details are illustrated in Figures 4, 5, and 6, respectively.

B Statistical Metrics Details

The specific definitions of the statistical metrics ϕ used to construct both single-dataset and double-dataset rules are summarized in Table 6.

C Proof of the FPR Upper Bound

We prove the upper bound on the expected false positive rate (FPR) stated in Eq. (4) using Chebyshev's inequality.

For any random variable X with mean μ and variance σ^2, Chebyshev's inequality states that

$$P\left(|X-\mu| \geq k\sigma\right) \leq \frac{1}{k^2}, \quad \forall k \in \mathbb{R}^+.$$

Let $\phi(D, C)$ denote the statistic computed on a set of columns C over a randomly sampled dataset D. Since RegValidator is training-free and does not access ground-truth labels, and anomalies are typically rare in real-world data, we treat randomly sampled datasets as approximately anomaly-free, which is a common trick in anomaly detection.

Given sampled datasets $D_{\text{sample}} = \{D_1, \ldots, D_t\}$, we estimate the mean μ and variance σ^2 of $\phi(D, C)$. For a tolerance parameter $\beta > 0$, the rule R flags C if $\phi(D, C) \notin [\theta_{\text{low}}, \theta_{\text{up}}]$, where $\theta_{\text{low}} = \mu - \beta$ and $\theta_{\text{up}} = \mu + \beta$.

Applying Chebyshev's inequality with $X = \phi(D, C)$ and $k = \beta/\sigma$, we obtain

$$P\big(|\phi(D, C) - \mu| \geq \beta\big) \leq (\sigma/\beta)^2 .$$

The event $|\phi(D, C) - \mu| \geq \beta$ is equivalent to $\phi(D, C) \notin [\theta_{\text{low}}, \theta_{\text{up}}]$, which corresponds to the condition under which rule R flags a set of normal data under the approximation. By definition, this probability corresponds to the false positive rate of R on columns C over dataset D.

Therefore, for any set of columns C over dataset D, the false positive rate is bounded by $P(\text{FPR}(R)) \leq (\sigma/\beta)^2$. Taking the expectation over the randomness of the dataset sampling, we obtain

$$\text{E}[\text{FPR}(R)] \leq (\sigma/\beta)^2 .$$

D Proof of Theorem 1

The proof of the approximation guarantee for the greedy approximation algorithm is adapted from Theorem 3 in the work on the Budgeted Maximum Coverage problem (Khuller et al., 1999). We provide a detailed derivation in the context of rule selection under an FPR budget.

Let $\text{OPT} \subseteq \mathbf{R}$ denote the optimal set of rules satisfying the budget constraint δ. Let r be the iteration in the greedy algorithm when the first element of OPT is considered but cannot be added due to the budget limit. Let l denote the number of elements added to V in the previous r iterations.

For clarity, we renumber the rules selected by the algorithm so that R_i denotes the i-th rule added to V, $i = 1, \ldots, l$, and let R_{l+1} denote the first element from OPT that could not be added. Define the partial sets $V_i = \bigcup_{j=1}^{i} R_j$, for $i = 1, \ldots, l+1$. Let j_i denote the index of the iteration in which R_i is considered. The coverage of R_i denoted by W_i',

Dataset	Regulation	Authority
Healthcare	Medicare Fraud & Abuse: Prevent, Detect, Report	Centers for Medicare & Medicaid Services
	Fiscal Year 2023 Medicare and Medicaid Integrity Programs Report to Congress	Centers for Medicare & Medicaid Services
	The False Claims Act: A Primer	U.S. Department of Justice
FDTD & Credit Card	The FATF Recommendations	The Financial Action Task Force
	FinCEN Suspicious Activity Report (FinCEN SAR) Electronic Filing Requirements	Financial Crimes Enforcement Network
	Indicators of suspicious activity for the banking sector	Australian Transaction Reports and Analysis Centre
	Payment Card Industry Data Security Standard: Requirements and Testing Procedures, v4.0.1	PCI Security Standards Council

Table 5: Regulations used in three datasets.

Type	Metrics	Definition
Single-dataset	*min*	defined as the minimum value in a numeric column
	max	defined as the maximum value of a numeric column
	mean	defined as the arithmetic mean value of a numeric column
	median	defined as the median value of a numeric column
	range	defined as the difference between the maximum and minimum values in a numeric column
	row_count	defined as the total number of rows in a column
	unique_ratio	defined as the number of distinct values divided by the total number of entries in a column
Double-dataset	*JS_div*	Jensen–Shannon divergence (Endres and Schindelin, 2003), defined as a symmetric and normalized variant of Kullback–Leibler divergence for measuring similarity between two probability distributions
	Cohen_dist	Cohen's d (Cohen, 2013), defined as a standardized effect size that quantifies the mean difference between two samples relative to their pooled standard deviation
	L-inf	L-infinity distance (Chebyshev distance) (Cantrell, 2000), defined as the maximum absolute difference between corresponding elements of two vectors
	KL_div	Kullback–Leibler divergence (Endres and Schindelin, 2003), defined as an asymmetric measure that quantifies how one probability distribution diverges from a reference probability distribution
	KS_statistic	Kolmogorov-Smirnov statistic (Smirnov, 1939), defined as the maximum absolute difference between the empirical CDFs of two samples

Table 6: Statistical metrics used for rule generation.

defined as $W_i' = |\text{SatiSet}(R_i)|$. The coverage of a rule set V denoted by $w(V)$, defined as $w(V) = |\bigcup_{R_i \in V} \text{SatiSet}(R_i)|$, and the total cost of V is denoted by $c(V)$, defined as $c(V) = \sum_{R_j \in V} c_j$. Let W_i denote the marginal gain of R_i, defined as $W_i = |\text{SatiSet}(R_i) \setminus \bigcup_{R_j \in V i-1} \text{SatiSet}(R_j)| = w(V_i) - w(V_{i-1})$. Let c_i denote the cost of R_i, where $c_i = \text{FPR}(R_i)$.

Lemma 1

After each iteration j_i, $i = 1, \ldots, l+1$, the following holds:

$$w(V_i) - w(V_{i-1}) \geq \frac{c_i}{\delta}\big(w(\text{OPT}) - w(V_{i-1})\big).$$

Proof. Consider the marginal gain of R_i selected by the greedy algorithm. By the greedy choice, the marginal coverage gain to cost ratio of R_i is maximal, i.e., for any $R \in \text{OPT} \setminus V_{i-1}$,

$$\frac{w(R \cup V_{i-1}) - w(V_{i-1})}{c(\{R\})} \leq \frac{W_i}{c_i}.$$

Since the total cost of rules in $\text{OPT} \setminus V_{i-1}$ is bounded by the budget δ, we obtain

$$\frac{w(\text{OPT} \setminus V_{i-1})}{\delta} \leq \frac{W_i}{c_i}.$$

Using $w(\text{OPT}) - w(V_{i-1}) \leq w(\text{OPT} \setminus V_{i-1})$, the inequality follows.

Lemma 2

After each iteration j_i, $i = 1, \ldots, l+1$, the following holds:

$$w(V_i) \geq \left[1 - \prod_{k=1}^{i}\left(1 - \frac{c_k}{\delta}\right)\right] w(\text{OPT}).$$

Proof. We prove the lemma by induction on i, considering the greedy selection of rules R_i.

Base case ($i = 1$): At the first iteration, we have $w(V_1) = W_1' = W_1$, where W_1 denotes the marginal coverage of the first selected element. By the greedy rule, its unit-cost gain is maximized, i.e., $W_1/c_1 \geq w(\text{OPT})/c(\text{OPT})$. Since $c(\text{OPT}) \leq \delta$, this implies:

$$W_1 \geq \frac{c_1}{\delta} w(\text{OPT}),$$

which establishes the claim for $i = 1$.

Inductive step: Assume that after iteration $i-1$,

$$w(V_{i-1}) \geq \left(1 - \prod_{k=1}^{i-1}(1 - \frac{c_k}{\delta})\right) w(\text{OPT}).$$

By Lemma 1, the marginal gain at iteration i satisfies

$$w(V_i) - w(V_{i-1}) \geq \frac{c_i}{\delta}\big(w(\text{OPT}) - w(V_{i-1})\big).$$

Thus,

$$\begin{aligned} w(V_i) &= w(V_{i-1}) + [w(V_i) - w(V_{i-1})] \\ &\geq \left(1 - \frac{c_i}{\delta}\right) w(V_{i-1}) + \frac{c_i}{\delta} w(\mathrm{OPT}) \\ &\geq \left(1 - \frac{c_i}{\delta}\right) \left(1 - \prod_{k=1}^{i-1} \left(1 - \frac{c_k}{\delta}\right)\right) w(\mathrm{OPT}) \\ &\quad + \frac{c_i}{\delta} w(\mathrm{OPT}) \\ &= \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_k}{\delta}\right)\right) w(\mathrm{OPT}), \end{aligned}$$

completing the induction.

Concluding

First, consider a sequence $a_1, \ldots, a_n \in \mathbb{R}^+$ such that $\sum_{i=1}^{n} a_i = \gamma A$, for $A, \gamma > 0$. It is well known that the function

$$1 - \prod_{i=1}^{n} \left(1 - \frac{a_i}{A}\right)$$

is minimized when $a_1 = a_2 = \cdots = a_n = \gamma A/n$, in which case it attains the value $1 - (1 - \gamma/n)^n$.

For iteration J_{l1}, by applying Lemma 2 , we get

$$\begin{aligned} w(V_{l+1}) &\geqslant \left[1 - \prod_{k=1}^{l+1} \left(1 - \frac{c_k}{\delta}\right)\right] \cdot w(\mathrm{OPT}) \\ &\geqslant \left[1 - \prod_{k=1}^{l+1} \left(1 - \frac{c_k}{c(V_{l+1})}\right)\right] \cdot w(\mathrm{OPT}) \\ &\geqslant \left[1 - \left(1 - \frac{1}{l+1}\right)^{l+1}\right] \cdot w(\mathrm{OPT}) \\ &\geqslant \left(1 - \frac{1}{\mathrm{e}}\right) \cdot w(\mathrm{OPT}). \end{aligned}$$

Let $w(\{R_t\})$ denote the weight of the single rule R_t selected as the second candidate solution by the algorithm. By construction, $w(V_l) + w(\{R_t\}) \geq w(V_{l+1})$, and therefore

$$\max\{w(V_l), w(\{R_t\})\} \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) w(\mathrm{OPT}).$$

Hence, the final greedy solution satisfies

$$w(V_{\text{greedy}}) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) w(\mathrm{OPT}),$$

establishing the claimed approximation guarantee.

E Domain specificity and limited generalization

We present additional experiments to evaluate the robustness of RegValidator under incomplete regulation settings and its generalization to datasets without regulations.

Coverage	AUC-ROC	F1
100%	0.85	0.47
75%	0.85	0.45
50%	0.86	0.36
25%	0.80	0.35
0%	0.78	0.33
AnoLLM (SOTA)	0.649	0.033

Table 7: Results on RTAD-Creditcard under different regulation coverage levels.

Method	AUC-ROC	F1
Iforest	0.692	0.251
PCA	0.692	0.266
KNN	0.738	0.291
ECOD	0.692	0.282
DeepSVDD	0.713	0.258
RCA	0.727	0.320
SLAD	0.714	0.285
GOAD	0.717	0.295
NeuTral	0.681	0.195
ICL	0.719	0.298
DTE	0.714	0.239
REPEN	0.724	0.306
AnoLLM	0.712	0.279
RegValidator	**0.738**	**0.325**

Table 8: Results on seismic dataset with no regulations.

Robustness to Missing or No Regulations. To evaluate whether RegValidator overfits to fully specified regulations, we systematically vary the regulation coverage on RTAD-Creditcard from 100% to 0%. Under this circumstance, the ideation agent is modified to generate anomaly ideas based on data information as well, rather than relying solely on regulations.

As shown in Table 7, RegValidator remains robust even when regulations are entirely removed. With 0% regulation coverage, it still achieves an AUC-ROC of 0.78 and an F1 score of 0.33, better than the SOTA.

Additional Experiment on a Dataset of Other Applications. We further evaluate RegValidator on the seismic dataset (Sikora and Wrobel, 2010), which is from another application and focuses on predicting whether seismic bumps are dangerous. Since this dataset contains no regulations, we applied our method in a no-regulation setting.

As shown in Table 8, RegValidator achieves an AUC-ROC of 0.738 and an F1 score of 0.325, which is the best overall performance.

F Hyperparameter Sensitivity Analysis

FPR Budget Sensitivity. We conduct a sensitivity analysis on the RTAD-Creditcard by varying the budget from 0.001 to 2.0. The results are reported in Table 9. We analyze the sensitivity of RegValidator to the false positive rate (FPR) budget used in the rule generation agent. Specifically, we vary the FPR budget from 0.001 to 2.0 on the RTAD-Creditcard.

FPR Budget	F1
0.001	0.4093
0.01	0.4568
0.1	0.4572
0.4	0.4572
0.8	0.4572
2	0.3807

Table 9: Sensitivity analysis of the FPR budget on RTAD-Creditcard.

As shown in Table 9, the performance is sensitive to extreme budget values. When the budget is too small, the validation becomes overly restrictive, leading to the removal of informative detection results. When the budget is too large, more noisy candidates are retained, which degrades performance. In contrast, moderate budget values yield stable performance. Therefore, we set the FPR budget to 0.4 in all experiments.

Convergence Analysis of Iterative SQL Generation. We provide a convergence analysis of the iterative SQL generation process in RegValidator to examine performance across multiple refinement iterations.

We evaluate the performance at different iteration steps on three datasets. As shown in Table 10, the average F1-score increases steadily from 0.58 (iteration 1–2) to 0.68 at iteration 5, after which improvements become marginal (≤ 0.01). AUC-ROC follows a similar pattern and stabilizes around iteration 3–5. Based on this observation, we select five iterations as the stopping criterion.

G Computational Efficiency and Cost Analysis

We have added detailed efficiency statistics between RegValidator and AnoLLM on three datasets. As shown in Table 11, RegValidator consumes 40K–155K tokens per dataset. The LLM invocation time ranges from 57s to 285s, while database execution takes 9s to 580s. Under the current API pricing (e.g., GPT-4o-mini at \$0.6 per 1M tokens), the estimated cost per dataset is \$0.02–\$0.09.

For comparison, AnoLLM requires both model training and long inference time. Its training time is up to 11641s, and inference time is up to 62899s.

H Case Study

Effectiveness of RegValidator. We first present additional case studies to further illustrate the effectiveness of RegValidator on different datasets.

On the RTAD-Creditcard dataset, the large data scale and high dimensionality make anomaly detection particularly challenging. Instead of judging anomalies solely based on individual attributes such as transaction amount, our method generates detection ideas that jointly consider multiple columns to capture transaction context. For example, when detecting abnormal transaction amounts, our method does not directly flag large amounts as anomalies. Instead, it combines customer identity and transaction category, and aggregates transactions within each category to determine whether the amount is abnormal under a specific context. This contextual reasoning enables our method to distinguish genuinely suspicious transactions from normal but high-value purchases, leading to more accurate anomaly detection.

On the RTAD-Healthcare dataset, our method extracts several effective detection ideas from regulations, such as identifying claims from beneficiaries who are not eligible for high-cost treatments, flagging providers with unusually high claim volumes, and reviewing claims submitted by beneficiaries with limited insurance coverage. These detection ideas are particularly effective for large providers with substantial claim activity. By generating SQL queries based on these ideas, our method is able to retrieve a comprehensive set of abnormal claim records, achieving more complete anomaly coverage compared with baseline methods.

Effectiveness of the data validation method. We also provide additional examples to demonstrate how the data validation method works.

On the RTAD-Healthcare dataset, one detection idea focuses on identifying anomalies by analyzing the number of claims submitted by providers. Based on this idea, our method first retrieves several candidate anomalous samples using an SQL query. Based on this idea, the rule generation agent produces a validation rule, namely the *unique_ratio* rule, which measures the ratio of unique providers to the total number of claim records needed below the threshold. For anomalous cases, a small number of providers submit an unusually large number of claims, resulting in a low *unique_ratio* (0.0006). In contrast, for normal data, claims are more evenly distributed across providers, resulting in a higher ratio, with a threshold of 0.4756. By comparing the retrieved samples with the overall data distribution, this validation rule successfully verifies the correctness of the detection idea.

On the RTAD-Creditcard dataset, another detection idea suggests examining transactions at distant merchant locations to identify potential fraud by cross-referencing the geographical data with the credit card holder's registered address. An SQL query was generated to flag transactions as anomalous if the difference in either latitude or longitude exceeds a threshold. The rule generation agent then produces a validation rule using the Jensen-Shannon (JS) divergence to compute the anomaly measure. Applying our validation rule to the results of this detection idea, the JS divergence is calculated to be 0.0008, which is smaller than the threshold of 0.0058. As a result, this detection idea fails the validation, indicating that it does not correspond to true anomalies.

Failure Case. To provide a more balanced perspective, we analyze a representative failure case based on

	RTAD-Healthcare		RTAD-FDTD		RTAD-Creditcard		Average	
iteration	**AUC-ROC**	**F1**	**AUC-ROC**	**F1**	**AUC-ROC**	**F1**	**AUC-ROC**	**F1**
1	0.80	0.74	0.80	0.76	0.86	0.24	0.82	0.58
3	0.79	0.76	0.90	0.85	0.86	0.31	0.85	0.64
5	0.79	0.76	0.88	0.89	0.86	0.38	0.84	0.68
7	0.79	0.76	0.90	0.90	0.86	0.40	0.85	0.68
9	0.79	0.76	0.88	0.89	0.86	0.42	0.84	0.69

Table 10: Performance across different numbers of refinement iterations.

	RTAD-Healthcare	RTAD-FDTD	RTAD-Creditcard
RegValidator			
Number of tokens	155,057	40,693	147,283
LLM invoking time	285.19s	57.30s	260.66s
Database execution time	343.23s	8.86s	580.41s
AnoLLM			
Training time	11,641.40s	765.68s	11,186.11s
Inference time	62,898.94s	1,085.66s	39,445.24s

Table 11: Computational efficiency comparison between RegValidator and AnoLLM.

transaction geolocation. On the RTAD-Creditcard, the generated audit rule assumes that transactions occurring far from a cardholder's typical latitude and longitude may indicate suspicious behavior. Although the corresponding SQL passes all validation rules and is consistent with the intended regulation, it achieves an F1-score of only 0.0097, indicating limited detection effectiveness. This result suggests that, in this dataset, large geographic deviations are not strongly aligned with the ground-truth anomaly definition.

I LLM Prompts

The prompts used to guide the large language models (LLMs) in this work are listed below. Each agent in our framework corresponds to one subsection, which contains all prompts used by that agent and may invoke multiple LLM calls during its execution. Except for the domain-specific expert role description in the background, the prompt structures are consistent across agents. Each step processed by an LLM, denoted as $f_{\cdot}^{\text{LLM}}$ in the main text, corresponds to the prompts provided in a single subsection. We use RTAD-Creditcard as a running example to illustrate the prompt inputs.

I.1 Ideation Agent Prompts

Potential anomaly analysis prompt

showstringspaces
You are an anomaly detection analyst with over 20 years of experience in data analysis and anomaly investigation, with a strong background in financial fraud detection for transactional datasets.

Now there is a set of tabular data stored in the database, which is used for anomaly detection tasks.
The following is a description of the dataset, including the basic information of the dataset and explanations of each field:
{data_info}

Please analyze the dataset and generate potential anomaly points in the following format:

1. [Anomaly Type]: [High-level description of the anomaly point]
2. [Anomaly Type]: [High-level description of the anomaly point]
3. [Anomaly Type]: [High-level description of the anomaly point]
...

Requirements:
- Each anomaly point should include the anomaly type and High-level description, not specific thresholds or explicit category labels.
- Focus on anomalies that can be detected in the dataset, based on the available data information.
- Ensure each point is distinct and covers different aspects of potential anomalies.
- Prioritize generating anomalies that involve a SINGLE field or a group of STRONGLY RELATED fields. Avoid combining multiple unrelated fields from different business aspects in one anomaly point, keeping the anomaly logic simple and focused.

Anomaly Ideas Extract Prompt

showstringspaces

You are an anomaly detection analyst with over 20 years of experience in data analysis and anomaly investigation, with a strong background in financial fraud detection for transactional datasets.

I will perform anomaly detection on the dataset. Based on the following information, please generate specific anomaly ideas:

[Potential Anomaly Points]
{potential_anomalies}

[Retrieved Policy Documents with Associated Anomaly Points]
{retrieved_policy_info}

[Dataset Schema Information]
{data_info}

Requirements:
1. Generate exactly One anomaly idea for EACH retrieved policy document fragment (total 15 anomaly ideas).
2. Each anomaly idea must correspond to both:
   - The specific policy document fragment it was retrieved for.
   - The associated potential anomaly point.
3. When formulating the anomaly approach:
   - Design a fit-for-purpose verification method aligned with the anomaly and policy context.
   - Consider the overall characteristics of the dataset to identify deviations from reasonable ranges, without specifying exact numerical thresholds or specific category values.
   - Prioritize focusing on a SINGLE business aspect of anomaly detection, using either a single field or a group of STRONGLY RELATED fields. Avoid involving multiple independent fields from unrelated business aspects.
4. Format each anomaly idea as follows:
[Number]. Anomaly Idea: [Brief purpose description] + [Concise operational steps describing the general approach]
Policy Document: [Policy document name]
Clause: [Specific clause content]

Additional Requirements:
- For each anomaly idea, design a practical detection method that is feasible based on the provided dataset schema.
- Choose the simplest effective approach that aligns with the anomaly and policy context.
- Ensure the anomaly idea is actionable and focuses on the core detection logic without referencing specific data fields or discrete category limits.
- Maintain the structure and completeness of each output item (Anomaly Idea, Policy Document, Clause).

Output Example:
1. Anomaly Idea: Identify records with missing critical information. Verify by checking for incomplete data entries that fail to meet basic documentation requirements.
Policy Document: Data Integrity Policy
Clause: All patient records must have a valid unique identifier.

2. Anomaly Idea: Detect abnormally high service volume. Assess by comparing service frequency against the general range observed across the dataset to identify outliers.
Policy Document: Billing Compliance Guide
Clause: Services must be medically necessary and commensurate with patient demographics.

Notes:
- Output plain text without using markdown.
- Provide exactly 15 anomaly ideas, one for each retrieved fragment.
- Do not include summaries or explanations.
- Ensure each anomaly ide is feasible within the available dataset schema.
- Focus on general operational logic in the anomaly idea rather than specific data fields or exact thresholds.

Table Association Prompt

showstringspaces
You are an expert in database structure and financial transaction auditing. Your role is to precisely map audit ideas to the correct database tables and their most relevant fields for SQL query generation. You must only use table and field names provided in the database information "and avoid inventing or modifying any content. Crucially, you need to: 1. Identify ALL fields involved in the judgment logic of the audit idea (ensure completeness, no essential fields omitted); 2. Select only the MINIMAL set of tables and fields that exactly meet the judgment requirements of the audit idea (ensure precision and minimal scope, no irrelevant fields / tables added).

Your task is to determine the required database tables and fields for each audit idea, following the rules of completeness, precision, and minimal scope.

[Database Information]
1. Table Names and Types:
{table_info}

2. Table Structures (fields and descriptions):
{data_info}

[Audit Requirements]
1. Each audit idea must be linked to relevant tables / fields strictly from the provided database information (no invented / modified names).
2. Both Required Table Names and Required Fields must be provided for every audit idea.
3. Required Table Names:
   - Only include tables that are DIRECTLY and NECESSARILY related to the audit idea;

- Use the minimal set of tables (no redundant, inferred, or newly created tables).

4. Required Fields:
   - Include ALL fields involved in the anomaly-detection logic of the audit idea (ensure no essential fields are omitted);
   - Only include the minimal set of fields sufficient to fulfill the judgment requirements (ensure no irrelevant fields are added);
   - Fields must be strictly extracted from the provided table structures.
5. The number of elements in the output results must be exactly equal to the number of input Audit Ideas, and each element must correspond one-to-one to each Audit Idea in order; if no relevant tables or fields are found for an Audit Idea, the corresponding required_tables or required_fields must be output as an empty list [].

[Audit Ideas]
{audit_ideas}

[Output Format]
Output a JSON array where each element corresponds to an audit idea. Each element should be a JSON object with the following keys:
- "audit_idea": the text of the audit idea.
- "policy_document": the text of the policy document.
- "clause": the text of the clause.
- "required_tables": a list of strings representing the minimal set of table names required for the audit idea.
- "required_fields": a list of strings representing the complete and minimal set of field names (from the required tables) needed for the audit idea's judgment logic. If no fields are relevant for a table, use an empty list [].

[Example Output]
Return the results in JSON format with the following structure:

```
{{
  "results": [
    {{
      "audit_idea": "Evaluate transaction frequency patterns for individual credit card numbers, focusing on unusually high activity within a short time frame to identify possible card compromise or fraudulent use.",
      "policy_document": "PCI-DSS v4.0.1",
      "clause": "The risk related to allowing new members of staff access to the Cardholder Data Environment (CDE) is understood and managed.",
      "required_tables": ["CREDIT_CARD_TRANSACTIONS"],
      "required_fields": ["cc_num", "unix_time"]
    }},
    {{
      "audit_idea": "Conduct a geographical review of transaction locations against the registered addresses of cardholders to identify inconsistencies that may signal identity theft or unauthorized use.",
      "policy_document": "Fraud_Abuse_MLN4649244.docx",
      "clause": "A customer displaying inconsistencies in identification information may indicate suspicious behavior.",
      "required_tables": ["CREDIT_CARD_TRANSACTIONS"],
      "required_fields": ["lat", "longitude", "merch_lat", "merch_long"]
    }},
    {{
      "audit_idea": "Check for transactions with zero amount that are not marked as test transactions.",
      "policy_document": "Internal_Fraud_Policy.docx",
      "clause": "All test transactions must be clearly labeled and have zero monetary value.",
      "required_tables": [],
      "required_fields": []
    }}
  ]
}}
```

I.2 Query Agent Prompts

Query Generation Prompt

showstringspaces
You are an audit expert and SQL specialist capable of analyzing audit ideas and database schemas to generate ORACLE SQL queries.

Based on the required table names mentioned in each audit suggestion, link to the corresponding table content information, and generate ORACLE SQL statements with reference to the field type and description information in Key Table Structures.

Audit Idea and Table Suggestions (contains audit idea, policy document, clause, required table names, and required fields):
{idea}

Database Table Information (Includes table names and types):
{table_info}

Key Table Structures (Includes field names, descriptions, and field types for each table):
{data_info}

When generating ORACLE queries:
- For each audit idea, generate THREE SQL statements:
  1. The first one should apply **the strictest threshold** (identify only the

most likely anomalies).
2. The second should apply a **moderate threshold** (balanced anomaly detection sensitivity).
3. The third should apply **the loosest threshold** (detect broader potential anomalies).
- If the audit idea is not suitable for threshold variation, output three identical SQL queries.
- Maintain logical consistency and readability among the three queries. Threshold changes should reflect reasonable ranges of statistical metrics.
- Given that abnormal data is scarce, prioritize setting rigorous judgment conditions and thresholds overall—even for the "loose" threshold, maintain a relatively strict standard compared to general scenarios to avoid excessive false positives.

Core Rules for SQL Generation (Prioritize Statistical Judgments Over Exact Values/Categories):
1. Reference Key Table Structures to determine field types (numeric/categorical) and follow the corresponding judgment logic:
   - For NUMERIC fields: Avoid using exact numeric values for judgment.
   - For CATEGORICAL fields: Avoid exact category value judgments.
2. The query must return all fields, using SELECT *.
3. The SQL should be generated strictly according to the audit logic.
4. You must strictly follow the required fields provided in the Audit Idea and Table Suggestions to generate the SQL.
5. All conditions in the WHERE clause must use only the Required Fields specified in the audit logic—no additional fields allowed.
6. Keep the SQL simple and focused on the core audit detection logic, avoiding unnecessary complexity.

Additional SQL Generation Requirements:
- Ensure logical clarity and avoid full table scans to improve query efficiency.
- Guarantee accuracy; use exact table and column names as defined in the database. Do not fabricate or guess names.
- Strictly follow the provided audit idea, policy documents, and clauses without modification.
- MINIMIZE THE USE OF UNNECESSARY AGGREGATION OPERATIONS. Focus on retrieving raw data records that exhibit abnormal patterns (use window functions for statistical calculations to retain raw records instead of aggregating and summarizing results).
- Return search results in their original data format rather than aggregated summaries to allow for detailed review of exceptions.
- Each SQL query should reflect different detection strictness levels through adjustments to statistical thresholds.

Output Rules:
- If there are table names, generate the three Oracle queries as per the requirements.
- Output only the ORACLE query statements. Do not include any other information.
- Do not use markdown formatting.
- Ensure the SQL statement does not end with a semicolon ';'.
- If you feel that you cannot generate correct Oracle queries with the existing tables and recommended fields, output {{"sql1": None, "sql2": None, "sql3": None}}.

Return the results in JSON format with the following structure, Do not include anything else:
```
{{
  "strict": "SELECT ... FROM ...",
  "moderate": "SELECT ... FROM ...",
  "loose": "SELECT ... FROM ..."
}}
```

Query Refinement Prompt

showstringspaces
You are an expert Oracle SQL auditor and optimizer.
Your task is to REVISE each SQL query based on its specific validation feedback, while maintaining the relative strictness hierarchy.

CRITICAL ADJUSTMENT RULES:
For EACH query, apply EXACTLY the required adjustment based on its Required Action:

1. "Required Action: MODIFY": Fix syntax errors—completely rewrite the SQL to resolve database errors.
2. "Required Action: STRICT": Make WHERE conditions MORE restrictive than current version (increase thresholds ONLY).
3. "Required Action: LOOSE": Make WHERE conditions LESS restrictive than current version (decrease thresholds ONLY).
4. "Required Action: NONE": Keep the SQL unchanged.

STRICT CONSTRAINTS:
- ONLY modify the threshold values in WHERE conditions.
- DO NOT add, remove, or modify any fields in SELECT, FROM, JOIN clauses.
- DO NOT change the SQL structure (no adding/removing JOINs, subqueries, etc.).
- DO NOT alter the core logic beyond threshold adjustments.
- Maintain exactly the same field references and table relationships.

REVISION TASK: Regenerate three SQL queries for the audit idea. Each query must be adjusted based on its specific validation feedback.

AUDIT IDEA (contains audit idea, policy document, clause, required table names and required fields names):
{idea}

TABLE DESCRIPTIONS (Includes table names and types):
{ table_info }

KEY TABLE STRUCTURES (Includes key field names and descriptions):
{ data_info }

VALIDATION FEEDBACK & REQUIRED ADJUSTMENTS:
{review}

STRICTLY ALLOWED ADJUSTMENTS (ONLY THRESHOLD CHANGES):
For each SQL statement, make corresponding adjustments based on Required Action and Adjustment Suggestions, rather than its own Degree of Strictness .
When adjusting threshold values (whether for " STRICT" or "LOOSE" adjustments), make the changes as minimal as possible (e.g., small incremental/decremental changes to numeric values, 1–2 percentile points for percentiles , minor shifts to date boundaries).

For "LOOSE" adjustments (ONLY when the Required Action is LOOSE):
– Decrease numeric thresholds : "> 1000" –> "> 900", "> 0.9" –> "> 0.85", "> 5" –> "> 4", "> AVG + 2 * STDDEV" –> "> AVG + 1.5 * STDDEV".
– Change comparison operators : ">" –> ">=".
– Increase percentile values : "> 90th percentile " –> "> 85th percentile ".
– Widen date ranges by adjusting date boundaries .

For "STRICT" adjustments (ONLY when the Required Action is STRICT):
– Increase numeric thresholds : "> 700" –> "> 800", "> 0.8" –> "> 0.85", "> 2" –> "> 3", "> AVG + 1 * STDDEV" –> "> AVG + 1.5 * STDDEV".
– Change comparison operators : ">=" –> ">".
– Decrease percentile values : "> 80th percentile " –> "> 85th percentile ".
– Narrow date ranges by adjusting date boundaries .

KEY REQUIREMENTS:
– ONLY modify threshold values in WHERE conditions:
  1. For numerical fields (where " field_type " is marked as "numerical"), prioritize using fuzzy thresholds (e.g., mean, standard deviation , percentiles , etc .) to construct judgment logic .
  2. For categorical fields (where " field_type " is marked as " categorical " or " others "), explicit thresholds (specific category values or definite judgment criteria) can be used to construct judgment logic .
– The SQL should be generated strictly according to the audit logic .
– Fields referenced in the WHERE clause must only be the fields specified in "Required Fields " and "Key Table Structures ", no additional fields allowed .

PROHIBITED CHANGES:
– NO structural changes to SQL (no new JOINs, subqueries , etc .) .
– NO changes to aggregation functions or GROUP BY.

OUTPUT REQUIREMENTS
– Maintain identical FROM/JOIN structure and SELECT fields, using SELECT *.
– No trailing semicolons .
– After threshold adjustments , ensure : Query 1 >= Query 2 >= Query 3 in strictness .

OUTPUT FORMAT (EXACTLY):

Return the results in JSON format with the following structure exactly :
```
{{
  " strict ": "SELECT * FROM ... WHERE ...",
  "moderate": "SELECT * FROM ... WHERE ...",
  "loose ": "SELECT * FROM ... WHERE ..."
}}
```

DO NOT include explanations , comments, or markdown formatting.
DO NOT make any changes beyond threshold value adjustments .

I.3 Rule Generation Agent Prompts

Anomaly Selector Prompt

showstringspaces
You are a credit card data analysis expert skilled at generating data sampling queries , analyzing key fields for anomaly detection , and specifying appropriate anomaly generation strategies .

As a data analysis expert , analyze the provided credit card transaction data and generate sampling queries for comparative analysis . Your task is to logically derive key fields from the Audit Logic, identify the most critical field , and create a sampling query for Oracle databases .

[Dataset Information]
{ data_info }

[Audit Logic]
{ idea }

Based on the above, generate the following information logically :
1. **KeyFields**: A list of key field names derived from the Audit Logic (especially Required Fields). These fields will be used for multi–field association calculations in anomaly detection .

2. ** MostCriticalField **: The single most critical field for analysis and anomaly generation. This field MUST be selected from the KeyFields. If the audit involves multiple related fields, choose the field that best represents the core audit logic. This field should be the primary focus for anomaly detection. Anomalies will only be generated on this field, but KeyFields will be used for associated computations.

3. **Numeric**: A boolean value (True or False) indicating whether the MostCriticalField contains numerical data.

4. **Oracle Sampling Query**: A SQL query that samples data from the original dataset. **CRITICAL REQUIREMENTS:**
   - The query MUST SELECT ALL KeyFields explicitly.
   - The query should be syntactically correct for Oracle databases.
   - The query should perform random sampling, for example using ORDER BY DBMS_RANDOM.VALUE.
   - Do not use WHERE to filter the data.
   - Do not include a semicolon ';' at the end.
   - Do not specify sample size limits like ROWNUM or FETCH FIRST in the query.

5. **Anomaly Generation Approach**: The strategy for generating anomalies, which must be one of the following types:
- `unit_change`: Modify numerical values by scaling or setting extreme values. **For numerical fields only.**
- `data_volume_change`: Increase or decrease data volume to simulate unusual frequency patterns. **Applicable to any field type.**
- ` distribution_shift `: Alter distribution patterns of categorical data, potentially affecting multi-field associations. ** Applicable to any field type.**

FIELD CONSISTENCY REQUIREMENT:
- MostCriticalField MUST be one of the KeyFields.
- All KeyFields MUST be fields from the Audit Logic's Required Fields.

Return the results in JSON format with the following structure:

```
{{
    "KeyFields": [" list of field names that appear in the Audit Logic Required Column Names"],
    " MostCriticalField ": "name of a field that appears in the Audit Logic Required Column Names",
    "Numeric": true or false,
    "Oracle": "SQL query that selects all KeyFields",
    "AnomalyLogic": "One of: unit_change, data_volume_change, distribution_shift "
}}
```

RTAD-FDTD Data Information


```
{
  " dataset_info ": {
    "name": "Fraud Detection Transactions Dataset ",
    "dataset_name": "fdtd ",
    " description ": "Fraud Detection Transactions Dataset ( table : transactions ) – A synthetic financial
        transaction dataset designed for fraud detection and anomaly identification . Contains 21
        features including Transaction_ID, Transaction_Amount, Failed_Transaction_Count_7d, Risk_Score,
        and other fields . Suitable for fraud detection model training , anomaly identification , and
        risk analysis with numerical, categorical , and temporal data types .",
    "table_name": " transactions ",
    "sample_num": 33035
  },
  "column_info": [
    {
      "column_name": "Transaction_ID",
      " description ": "Unique identifier for each transaction ",
      "data_type ": "varchar ",
      "column_type": " others ",
      "format_examples": [
        "TXN_33553",
        "TXN_9427",
        "TXN_199"
      ]
    },
    {
      "column_name": "Transaction_Amount",
      " description ": "Amount of money involved in the transaction ",
      "data_type ": "double ",
      "column_type": "numerical ",
      "format_examples": [
        39.79,
        1.19,
        28.96
      ],
      "range ": [
        {
          "min": 0.0
        },
        {
          "max": 1174.14
        }
      ]
    },
    {
      "column_name": "Transaction_Type",
      " description ": "Type of transaction (Online, In–Store, ATM, etc.) ",
      "data_type ": "varchar ",
      "column_type": " categorical ",
      "format_examples": [
        "POS",
        "Bank Transfer ",
        "Online"
      ]
    },
    ...
  ]
}
```


Figure 4: RTAD-FDTD data information example.

RTAD-Healthcare Data Information


```
{
  " dataset_info ": {
    "name": "HEALTHCARE PROVIDER FRAUD DETECTION ANALYSIS",
    "dataset_name": " healthcare ",
    " description ": "The Healthcare dataset contains structured information about inpatient claims,
        outpatient claims, beneficiary demographics, clinical conditions, insurance coverage, and
        provider-level labels .",
    "table_name": " healthcare_data ",
    "sample_num": 385687
  },
  " field_info ": [
    {
      "field_name": "Provider ",
      " field_type ": " String ",
      "data_type ": " categorical ",
      " description ": "Unique provider identifier , links to fraud labels .",
      "format_examples": [
        "PRV56576",
        "PRV57113",
        "PRV56909"
      ]
    },
    {
      "field_name": "ClmAdmitDiagnosisCode",
      " field_type ": " String ",
      "data_type ": " categorical ",
      " description ": "ICD code for admission diagnosis .",
      "format_examples": [
        "V579",
        "72939",
        "55321"
      ]
    },
    {
      "field_name": "InscClaimAmtReimbursed",
      " field_type ": "Float ",
      "data_type ": "numerical ",
      " description ": "Amount reimbursed by insurance ",
      "format_examples": [
        4000,
        21000,
        84000
      ],
      "range": [
        {
          "min": 0
        },
        {
          "max": 125000
        }
      ]
    },
    ...
  ]
}
```


Figure 5: RTAD-Healthcare data information example.

RTAD-Creditcard Data Information


```
{
    " dataset_info ": {
    "name": " Credit Card Transactions Fraud Detection Dataset ",
    "dataset_name ": " creditcard ",
    " description ": "This is a simulated credit card transaction dataset containing legitimate and fraud
        transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000
        customers doing transactions with a pool of 800 merchants .",
    "table_name": " credit_card_transactions ",
    "sample_num": 652402
    },
    "column_info": [
        {
          "column_name": "trans_num",
          " description ": "Unique transaction number generated by the system to identify the transaction
              ",
          "data_type ": " varchar ",
          "column_type": " others ",
          "format_examples": [
            "6a10cbeefaaf7e089a7dad8ea8575276",
            "a1736841e877bb846764e19036504459",
            "323a30c15cf2ec3e0164ed8473473c87"
          ]
        },
        {
          "column_name": "amt",
          " description ": "Amount of the transaction , represented as a numeric value ",
          "data_type ": "double ",
          "column_type": "numerical ",
          "format_examples": [
            45.99,
            120.5,
            3.75
          ],
          "range ": [
            {
              "min": 1.0
            },
            {
              "max": 28948.9
            }
          ]
        },
        {
          "column_name": "category",
          " description ": "Category of the merchant's business ",
          "data_type ": " varchar ",
          "column_type": " categorical ",
          "format_examples": [
            "misc_net ",
            "grocery_pos ",
            " entertainment ",
          ]
        },
        ...
  ]
}
```


Figure 6: RTAD-Creditcard data information example.