ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

Handling Criteria-Driven Filtering &Sampling Across Distributed Data Partitions

Nagabhushanam Bheemisetty

Independent Researcher, USA

DOI: https://doi.org/10.51583/IJLTEMAS.2026.15020000122

Received: 27 February 2026; Accepted: 05 March 2026; Published: 23 March 2026

ABSTRACT

This paper presents an innovative five-layered architecture for finding equitable representation across the various fragmented sections of large datasets. The architecture employs a set of customizable criteria to group together similar datasets into well-balanced unbiased sample sets; while retaining complete and accurate metadata for all file boundary elements. Important features of this architecture include: a multi-level criteria engine which is capable of processing accepted and rejected streams from a variety of sources; a stratified sampling process which effectively eliminates the presence of partition skew; and a quality assurance mechanism that provides an efficient means to capture and report performance-related metrics. Benchmark partitioning results indicate a significant shift toward overall balance ratio improvements; with improvements of approximately 1.75 to 1.12 ratio, as well as an overall reduction in volume anomaly of 87% (from ±25% down to ±3.2%), and total absence of schema drift occurred at a cost of less than 5% of the overall dataset processing costs to the author(s). By implementing this method, the processing of large datasets has been made much more reliable than previously possible, while simultaneously minimising bias and maximising the ability to provide efficient audit trial capabilities. Future enhancements to the architecture will focus on providing distributed execution capabilities in conjunction with streaming integration into enterprise data systems.

Keywords: Partitioned Datasets, Balanced Sampling, Data Unification, Criteria Engine, File-Level Fairness, Anomaly Detection

INTRODUCTION

Once a multi-file dataset has been separated into multiple serial file segments, the goal of the system is to represent it as one dataset, rather than as many individual datasets. The idea behind this design is to ensure that analyses and sampling of datasets remain a cohesive unit, therefore eliminating the risk of any particular file becoming over or under-represented in the final output of the system. Having file-level balance during sampling is vital to ensuring that all serial segments contribute their fair share of information, therefore maintaining the integrity of the distribution and attributes originally contained within the multi-file dataset. In order to maintain the logical coherence and balance of all serial segments, the system must coordinate the functioning. When analyzing large amounts of data or when analyzing data across several separate locations, accurate sampling of data from a dataset is very important. To provide a more efficient way of sampling, many datasets are broken into segments based on how they will be used, how they will be moved, and for other reasons. [1]

By using a sample with balanced representation from each of the dataset's files, the sampling preserves many of the original characteristics of the dataset; that is, the original distribution of records should still be the same after sampling. For example, if three files contain different number of records—one has only ten records, another file has twenty records, and the last file has 30 records—when creating a balanced representation of 60 records from these files, the sample would contain ten records from the first file, 20 records from the second file, and 30 records from the last file. Therefore, the use of balanced representation prevents over-representation or under-representation of any one file, which could potentially distort the results of a study based on random selection of records across files. If multiple serial file segments are to be combined together into a single logical unit of data for further analysis, care must be taken to ensure that all files are formatted and contain the same format

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

when combined. Examples of formats would be the name of the columns, data type, and order that the file's records should be written.

The actual combining of the files into one file is generally done by concatenating each of the files' records in the order in which the serial files were listed. A number of applications, programming languages, and products exist that allow you to concatenate file records, examples include using pandas in Python, using pd.concat() function to append multiple DataFrame objects together, or listing the files and reading them into a DataFrame object using the glob module before analysis. There are also command line utilities available for combining text-based files, e.g. copy and/or cat utilities for basic text files and dplyr for structured data in R. Whereas IBM and JCL tools are commonly used for sequential data management in enterprise environments. Merging files into one cohesive dataset by implementing these procedures provides the ability for further analysis/sampling/modeling, etc. by creating a new dataset that can be sampled, analyzed, and/or modeled [2].

Often large datasets are separated into multiple parts of different file sizes for processing. When working with these files, naive processing of each part as an independent file may produce bias in terms of results produced. In response to this bias, a systematic process is developed to provide the ability to identify datasets with similar logical groupings, with each dataset containing sequentially written partitions that represent 'slices' of the dataset. The process consists of reading all partitions of a dataset and assembling them into a single table/DataFrame, utilizing customized keys to identify the assembled data. The assembled data can now be subjected to comprehensive filtering rules that are applicable to all three or more files made up of these same logical partitions, providing assurance that identified cross-file combinations do not become lost.

In order to provide a balanced sample to be used for the purpose of statistical analysis/modeling/sampling, the process will trace back through the files to identify the original partitions from which the samples were drawn, allowing for representative sampling based on proportional representation. For example, if we compared this process of representative sampling from files to sampling various fruits from multiple bags without regard to the types/quantity of fruit within each bag, if we sort the bags and equally represent each bag it removes the opportunity for bias in the analysis. The significance of this process is twofold: it reduces the risk of undersampling from critical partitions; it allows for advanced management of large-scale datasets for the purpose of performing sampling/analysis/modeling, etc.; it also allows for the use of configurable parameters to meet business-specific needs.AI and machine learning workflows can be both equitable and adaptable to the underlying data in the logical data model (LDM) and will provide insight into and improve productivity in a fair manner.

In most cases, the amount of data processed each day by organizations within ecommerce is very high, resulting in individual transaction log files that are too large to store on one piece of storage media (e.g., disk drive) at any single time. In addition, many ecommerce businesses use concurrent processing (i.e., processes run simultaneously) for the different transaction-files, and so the files typically have multiple filenames (one per day). This makes it possible for the organization to analyze daily transactions as if the organization has one file for day-by-day analysis, without being limited to analyzing any one transaction log file alone. To illustrate this, consider an organization wanting to generate a report listing high value customers. To do this, the organization would read from all of the transaction log files (the "daily transaction log files") during the time period of interest to create one combined dataset to which they would then apply a common set of filters and selection criteria so that all datasets within the combined data would provide a level of contribution equal to one another. Following this method will allow the organization to analyze a sample that provides an equal level of contribution from all source datasets listed in the report, no matter how or when the original log files were created [4].

To create an equitable and consistent level of contribution to data processing for each file after file ingestion, it is important to accurately account and adjust for contribution levels as file parts are ingested. Contribution levels can be determined by monitoring the amount of data in each file to determine whether the volume of data for any portion is disproportionately large in comparison to the total volume of data ingested. The following are examples of methods used to create equitable data contribution levels for each file portion:

(1) monitoring the volume of data,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

(2) analyzing the summary statistics of each file portion to determine which summary statistics exhibit significant variances, and
(3) Utilizing proportionality tests to determine if any file portion exhibits a dominant level of contribution to the overall dataset.

Corrective actions taken when creating equitable data contribution levels for individual file portions include

(1) resampling,
(2) stratifying the sample to ensure equal representation, and
(3) utilizing data augmentation when necessary to account for under-representation of components based on business justification.

The workflow for equitable processing involves collecting and documenting metadata about all parts of the file, analyzing the contribution level and distribution patterns of all parts, applying resampling or stratified sampling methods to rectify imbalances, and ensuring that an equal and representative sample is prepared prior to processing the data. Each of these approaches helps mitigate the effect of any imbalance that may exist at the part level and maintain the integrity and objectivity of the analysis of partitioned (i.e., segregated) datasets [5].

To ensure a balanced representation of each file section after initial ingestion, several methods exist. For example, the proportional adjustment method ensures that each file section is distributed to an appropriate representative sample level equal to its size/proportion in the dataset. The threshold-based rebalancing method ensures that each component is monitored and that appropriate steps are taken if thresholds are exceeded. The periodic rebalancing method ensures that the contribution levels of file sections are evaluated at regular intervals and adjusted as necessary to maintain a balanced representation. The hybrid approach includes both threshold-based rebalancing and periodic assessments of rebalancing to adjust as needed and at specific intervals. These methods enable organizations to prevent the introduction of bias due to the effect of imbalances [6].

Related Work

In published literature, there are different methods for performing repartitioning on large data sets that have been partitioned into smaller subsets. Each approach offers its own advantages and disadvantages along with various challenges associated with that particular method. R-Grove is one method where an enhanced R*-tree index is used to generate partitioned areas and take into account user-defined balancing factors (e.g., weight ratio). This method allows for both load distribution and the ability to maintain spatial quality through the use of size histograms and example points. Another method, SABBS/SABBSR, focuses on load distribution and maintaining a high level of data locality when accessing a distributed memory system. Use of the two-stage heuristic algorithm provides a way to analyze and determine optimality and trade-offs. It allows for improved load distribution and processing times using a division-based redistribution process. Still, there are scaling and local data issues with respect to dynamic workloads. And, while all the methods discussed above can effectively redistribute large-scale datasets that have been partitioned in accordance with a site type and/or environment the main benefits are relative to the site type and/or environment where the dataset will be processed [7][8].

An example of an effective option for managing geographic data that has been partitioned under a distributed environment is R-Grove. An example of a method that will allow for improved geographic locality when performing query-based searches is SABBS/SABBSR. The contribution toward efficient spatial data analytics using incremental partitioning as a framework, which allows for a continuously maintained set of spatial partitions that perform better than static approaches through incremental partitioning. A separate study discusses different strategies for distributed spatial data analytics using MapReduce, providing insight into the competing trade-offs associated with query performance and scalability. While other works have focused on partitioning spatiotemporal data, the goal of these studies is to provide a balanced load to geographically distributed clusters through various partitioning approaches. Partitioning is typically done through bucket-based and tree-based methods. Each method has its pros and cons; i.e., the high price of processing, frequent, and large amounts of overhead due to reconstruction of partitions [9][10].

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

Numerous studies now provide extensive benchmarking of distributed spatial partitioning strategies in distributed systems, i.e., on Hadoop/Spark/Spark SpatialHadoop, and provide evaluations of various performance metrics that evaluate the effectiveness of each method. For example, the work of Aji, Kairey, and Xu (2015) evaluated six partitioning methods on larger MapReduce datasets and showed that STR and Quad-tree methods provide better scalability and query response times, and load-balancing capabilities. The study by Zeidan and Vo (2022) also makes a significant contribution to spatial partitioning accuracy benchmark development by comparing mapping and kNN joins across a range of different application datasets on Hadoop/Spark. The work of Niopas (2018) audited the relative performance of R/R+-tree and Quad tree delivery models in Hadoop File System (HDFS) workloads. The audit concluded that quad trees outperform other delivery models for a majority of workloads.Moreover, Zeidan and Vo's research on Efficiently Partitioning Spatial Data for Distributed kNN Joins shows that by developing novel partitioners for their data, it is possible to minimize the level of data skew and improve the performance of kNN joins through improved load balancing, as shown by the results of this benchmarking effort. The benchmarks use real-world GIS datasets and include metrics such as load balancing ratio, query throughput, and workload variation. Overall, these studies illustrate how to weigh the benefits of balanced versus efficient querying and provide guidance for researchers choosing the right partitioning strategy for Distributed Spatial Analytics [11][12][13].

The real-world GIS datasets that are commonly used in Spatial Partitioning Research are typically sourced from the TIGER/Line, OpenStreetMap, and Federal repositories and include road networks and points of interest (POIs). They are used as the basis for testing queries, assessing load balancing of distributed systems (such as Hadoop and Spark), and evaluating skew in data. The TIGER/Line dataset consists of millions of road segments, which are commonly used as benchmarks for partitioning and range query performance. The North American Road and POI datasets provide a basis for load balancing tests. The NYC Taxi Trips dataset is useful for testing spatiotemporal partitioning. The OpenStreetMap dataset contains the world's building and road networks and can be used to evaluate scalability. These datasets have been referenced in many papers, demonstrating their usefulness in evaluating various partitioning techniques like Quad-trees, STR, and Hilbert curves based on their availability to the public, and their ability to replicate results [14].

Synthetic datasets are also essential to Spatial Partitioning Research, as they allow researchers to evaluate algorithms under controlled conditions with varying densities, clusters, and degrees of skew and size. Examples of synthetic datasets include all points having equal density uniformly across a two or three-dimensional space, Gaussian mixtures having clustered densities, and skewed distributions such as Power-law or Zipf that simulate urban-rural environments. Multi-modal datasets can also be utilized to test partitioning on complex geometries. Many foundational papers in the field utilize these datasets to perform load balancing and skew management analyses, along with the non-convex partitioning problems found in many cases using large volumes of points, allowing researchers to model and simulate real-world conditions. Synthetic datasets improve the credibility of studies through the ability to replicate outcomes under extreme conditions through the use of custom spatial samplers or the use of scikit-learn's make_blobs for their creation.

Synthetic datasets are frequently used in partitioning studies to analyze the scalability, skewness, and load balancing of distributed systems. Synthetic datasets generated through the use of reproducible parameters in the scikit-learn package provide the means by which controlled distributions can be employed in evaluating the scalability, skewness, and load balancing of Distributed Systems.

The typical configurations will vary depending on the dataset used; for example, uniform distributions will have even densities throughout the defined area, but with fixed seeding parameters. The parameters of the Gaussian mixtures (clustered distributions) will define the positions of the clusters and the standard deviations of each cluster; thus, they will control both tightness and spread of the cluster distributions.

The uniform or skewed distribution types (Zipf or Power-law) will create hotspots, wherein a large percentage of points will congregate. The key parameters will be referenced in most papers, while allowing manageable complexity through maintaining consistent random states. Often, testing Distributed Systems will involve a range from 10 million points to greater than 1 billion points.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

System Architecture

The Data Processing Framework is a system that integrates multiple datasets by their business keys into one common DataFrame; however, the data processing framework maintains file boundaries when samples are being collected or filtered but will allow for Records to continue through multi level filtering engine based on filters applied when routed. The balancing engine uses a method of proportional allocation to ensure that representative samples represent each part of the data divided by the business key and retain file representative characteristics; rounding issues are resolved within the balancing engine. The Output Layer for the Data Processing Framework provides both Acceptable Samples that can be used for ML, as well as reject samples which may be used for compliance auditing purposes, allowing for full traceability. There are several major Benefits to this Data Processing Framework including but not limited to: scalability for large datasets, preserving distributions through proportional sampling, audit readiness and high flexibility in terms of the filtering criteria used to sort sample records through the multi-level filtering engine. This architecture allows fragmented unstructured or semi-structured data to be transformed into reliable samples for support of artificial intelligence/ machine learning, analyses, and compliance; thereby, reducing the risk of loss of integrity or accuracy due to the uneven partitioning previously shown in Figure 1:

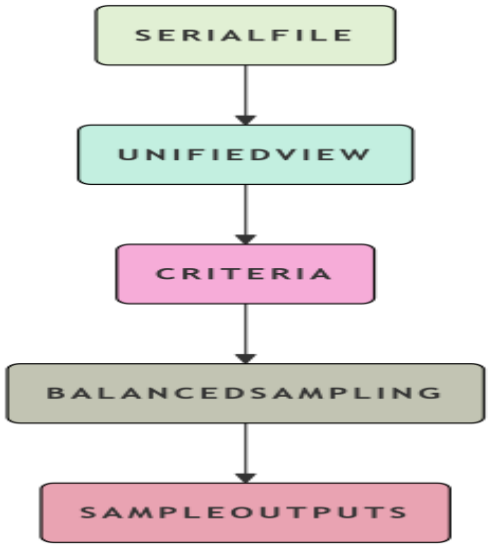


Figure 1:

This document consists of an overview of the 5 major layers through which a unique dataset will be created from fragmented datasets, each consisting of multiple partitions; sales files will represent one category of data as an example (not the only example). It does not represent an exhaustive list of the layers of the pipeline. It is, however, designed to provide readers with a logical flow through the end-to-end process of creating a unified dataset from the fragmented datasets described above.

Serial File Parts Input Layer:

- This layer is focused on the visibility of datacorp actual files for sales files in multiple parts.
- This layer will generate a raw file stream, and a Metadata Output, which will be generated at the Physical Data division and Data Ingestion Batch configuration.
- Fragmented inputs will be treated as logical groups (example: Dataset A consisting of multiple parts).

Consumption & Integration Layer:

- This layer will read-through all segments of the physical file segment and logically combine them through the use of business keys (transaction_id, timestamp, customer_id).
- A DataFrame will be created from the combined output of this layer (consumption and integration) and will allow you to trace back to the source of the data and perform Sampling & Filtering operations.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

Filtering Criteria Engine Layer:

- The filtering criteria engine identifies whether or not an input record meets a particular Filtering Criteria, using configurable ROUTING through a set of business rules defined in JSON/YAML format.
- Records which are accepted will be available in the "Accepted Record" stream, while rejected records will be in the "Rejected Record" Stream. Both Accept and Reject streams can be populated by either JSON/YAML format.

Balanced Sampling Layer:

- The Balanced Sampling layer provides a Balanced Sample of all original sources of the Multi-Part Dataset.
- The output of the Balanced Sampling Layer is A Balanced Sample of the Original File Distribution.

Multiple Stream Output Layer:

- The Multiple Stream Output Layer is designed to support the development of Customized Solutions for Customers based on the diverse Use Cases associated with Accepted Records (Analytics Training) and Rejected (Compliance & Audit Records).
- Each Rejection will also be documented to provide full visibility into the Multiple Stream Outputs and provide Customers with the Knowledge and Tools necessary to utilize their records effectively.

The proposed Data Warehouse Architecture is built to process Partitioned Datasets by converting Raw Serial File components into Balanced Analytical Samples using Five Different Layers. Each layer serves a unique purpose and supports the overall process as illustrated below. The first layer is the Serial File Parts Data Source Layer. In this layer you will find multiple systems that have collected disjointed inputs. Each input has been logically grouped together but is not Schema Enforced; therefore we cannot assume that one input is equivalent to another.

The second layer is the Staging/Data Integration Layer. This layer is used to join all of the disjointed inputs together by means of an ETL-type process. The inputs that are collected in the first layer are extracted, joined together based on a Business Key(s) and then normalized. All of these input files are combined into a single Unified Working View. The Unified Processing View (Data Storage Layer) comprises the third layer of the Data Warehouse Architecture. The Unified Processing View is a Central Place for the Business Logic and is structured as a Dimensions Based Fact Table. The Purpose of this layer is to allow for efficient querying on the Business Facts while preserving the boundaries of the input files.

The Analytics and Transformation Layer is the fourth Layer and uses a Criteria Engine, to support Balanced Extraction of Sample Data based on the Business Rules. The Analytics Layer is designed to support Proportionally Balanced Sample Selection based upon established Business Rules. The Multi-Stream Outputs Presentation/Consumption Layer represents the Fifth Layer of the Data Warehouse Architecture. The final Output(s) from this Layer provide an Audit Ready and Actionable Result - separate valid and rejected Auditing Samples offered in CSV or Parquet Format and Full Metadata Lineage. This Architecture uses the Standard Data Warehouse Layers, but is modified to allow for the minimization of Partition Bias and the Maximization of Representative Sampling Across Multiple Sources; taking advantage of the Maximize Efficiency and Efficacy of Data Processing.

The Staging Region is part of the Ingestion and Unification of the Samples by consolidating All Fragmented Serial File Sections into a Cohesive Logical Dataset.The design begins to create an input and staging framework. The framework typically defines the following: (1) input datasets to be staged; (2) the schema and metadata associated with the respective datasets; (3) where the datasets will be stored; and (4) a method to calculate the size of the stage area (staging area-sizing formula). Once the datasets are defined, created, and prepared to be loaded into a staging area, a temporary staging table will be created to hold the attributes of the input datasets (staging.unified_view). As multiple input datasets have to be merged or transformed into one staging table, capacity planning (the size of the temporary staging table) must take place using a calculated sizing formula.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

Files are processed in parallel by a multi-threaded process which reads input files (from vfile or os) in parallel so that they can be processed simultaneously. Each input dataset is verified to have a matching schema, and the schema must be retained when capturing metadata while detecting and capturing errors during the metadata capture process.

This section describes the processes to be executed to enable the subsequent phases of executing joins and transforming the datasets into a unified view. All datasets in the workflow will have their data merged into one unified dataset and validated for completeness and quality prior to transferring the unified view output to the criteria engine. Metadata will be preserved regarding how the unified view was sampled, and how the unified view was filtered (what sampling process was used). Monitoring and observability of the entire staging process will take place via a dashboard that will provide key metrics related to how many parts were consumed, record count, join success rate, and time spent in staging. Overall, this product removes the physical fragmentation of data while preserving all required metadata, and facilitates balanced analytic processing.

The document presents a complete model for executing data quality validation in a multi-layered partitioned dataset workflow. The multi-layered approach allows for validation via the following input stage (data validity via completeness of data ingested and basic integrity checks: e.g., file and size completeness, schema matching, and record count) to determine whether data met the necessary requirements for further processing in the Data Staging and Unified View layers. Quality is preserved in the Data Staging and Unified View layers through joins of input datasets to form a unified view (staging layer) and maintaining the schema of input datasets and consolidated metadata. In the Data Staging and Unified View layers, quality metrics will be captured regarding the successful execution of joins and the discovery of duplicate records and parts. The Unified View layer will capture quality metrics regarding the data and the state (freshness) of the data. In the Criteria Engine layer, quality metrics will be captured regarding both data quality and completeness of filter criteria, rule applicability, and proportional preservation of criteria and rejected records. The Output layer will validate equivalency and representativeness of a sample. It will validate whether or not samples are proportional to requests, that they are random and that they have preserved their lineage (used in both the Criteria Engine and Output layers).

There are quality gates within the framework that are triggered to either send an alert or halt progression in the workflow, based on the level of issues detected (categorized into warning, error, and fatal). The monitoring dashboard will provide updates on the status of each layer, including alerts for critical failures and daily summary reports for warnings. A pseudocode template is provided for all quality checks, thereby permitting transformations to be executed in such a manner to ensure that data integrity is maintained and any issues related to partitioning are handled. Ultimately, through these methods, reliable samples will be generated for machine learning and analytics, as depicted in Table 1 below.

Layer	Focus Area	Key Checks	Critical Metrics	Thresholds	Failure Action
Input (File Parts)	Ingestion completeness	File presence, size, schema, record count	100% files, >1KB, schema match, ±10% volume	100% files found, identical schema	Halt pipeline, quarantine
Staging (Unification)	Join quality & metadata	Join rate, metadata completeness, duplicates, part balance	>95% match, 100% metadata, <0.1% dups, <30% max part	Metadata 100%, join >95%	Log unmatched, fail staging
Unified View (Storage)	Consistency & freshness	Freshness, volume anomaly, null rates, part weights	Within 24h, ±20% volume, <5% nulls, weights=1.0	Nulls <5%, weights ±0.01	Block criteria engine
Criteria Engine (Filtering)	Filter logic & stream integrity	Stream balance, rule coverage,	100% conservation,	Accepted + rejected = input	Detect data loss

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

		proportional preservation, reject logging	>0% rule hits, ±5% part ratio		
Output (Sampling)	Sampling fairness	Proportionality, size accuracy, randomness, lineage	±2% part ratios, exact size, high entropy, 100% lineage	Part diff <2%, lineage 100%	Resample immediately

Table 1: Data Quality Checks Across All Layers

The metrics that are associated with business value are based on data throughout the data pipeline and are focused on sampling fairness, sampling quality improvement, and reducing sampling bias. Key metrics in the process of evaluating data usage in the pipeline are sample partition balance, which ensures that no one part has more than its fair share of samples in the total sample, and sample proportionality, which shows whether or not the sample size is in proportion to the original distribution. The pipeline is evaluated based on quality gate metrics, which are in place to ensure that the data is fully available and has not been lost through the process of joining. The join success rate represents how well the data has been joined together and reflects how well a system can unify disparate data. In addition, business value is evaluated based on how well representative of the sample is, how efficiently the sample was processed, and how well the sample adhered to the defined standards for that company, with each metric having a specified threshold for adequacy. System performance metrics focus on system scalability, system latency, and memory efficiency of the system and indicate how well the system can support large amounts of data in an effective manner.

Governance metrics include audit and compliance, with a focus on complete traceability of each sample and quality checks on samples accepted into the system. The sample dashboard clearly shows a high level of processing efficiency, a balanced sample partitioning, and a complete understanding of the lineage of each sample within the sample. However, for a sample to be able to most effectively demonstrate its performance, all conditions for success must be achieved. The primary Key Performance Indicator (KPI) combines the two components of representativeness and partition balance. The ideal score for a KPI would demonstrate a high degree of fielity and balance and, therefore, show the user's confidence in the ability of the solution to provide reliable samples for use across many different uses. The metrics defined above are critical for determining whether or not there is bias, skew, or degradation of quality in data that has been partitioned and semi-structured in nature.

The partition balance ratio is one of the most important aspects of sampling for proportional success, and must not exceed 1.2 to ensure that one portion of the data will not dominate over others. The availability of partition volume must be less than 20% in order to recognize when unusual batching or ingestion occurs, and partitions must be updated on a daily basis so that stale data does not impact analysis results. The schema drift between the various partitions must remain 100% consistent to avoid unification errors; the variance of the null rates should remain within 5% to ensure fairness in sampling and allow for the identification of quality drops. Monitoring practices should include incremental checks of every partition separately, tracking trends for the last 90 days, and conducting ongoing monitoring of all partitions using the latest algorithms. The adjusted partition balance ratio will be the only key performance indicator for executives to review, as it must be above 0.85 in order to demonstrate fair sampling from the larger dataset, as described in the example above.

The datasets themselves can provide a wealth of information for analyzing different metrics related to fairness in sampling, proportionality error, and balance ratios across multiple domains. The New York City Taxi and Limousine Trips dataset has over 2.5 billion daily and weekly recorded journeys in Parquet format, and serves as an ideal testing vehicle due to its natural partitioning by date and weekend variances in the number of trips recorded each day. The ASR dataset from Librispeech contains audio splits and allows for the benchmarking of selection of subsets by demonstrating how balanced subsets reduce word error rates substantially. The Roads TIGER/Line dataset offers over 100 million segments and illustrates the effectiveness of quad-tree partitioning

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

in reducing skew within an area. The extracts from OpenStreetMap provide an avenue to conduct analyses of spatial data from various regions.

In a practical application, users may create a daily report of metrics obtained from the NYC Taxi dataset; calculate the balance ratios for the various partitions; and identify any volume anomalies within the dataset. They may also compare their results against Industry Standards that have been established for the previous two years, as demonstrated in the following Figures 2 and 3.

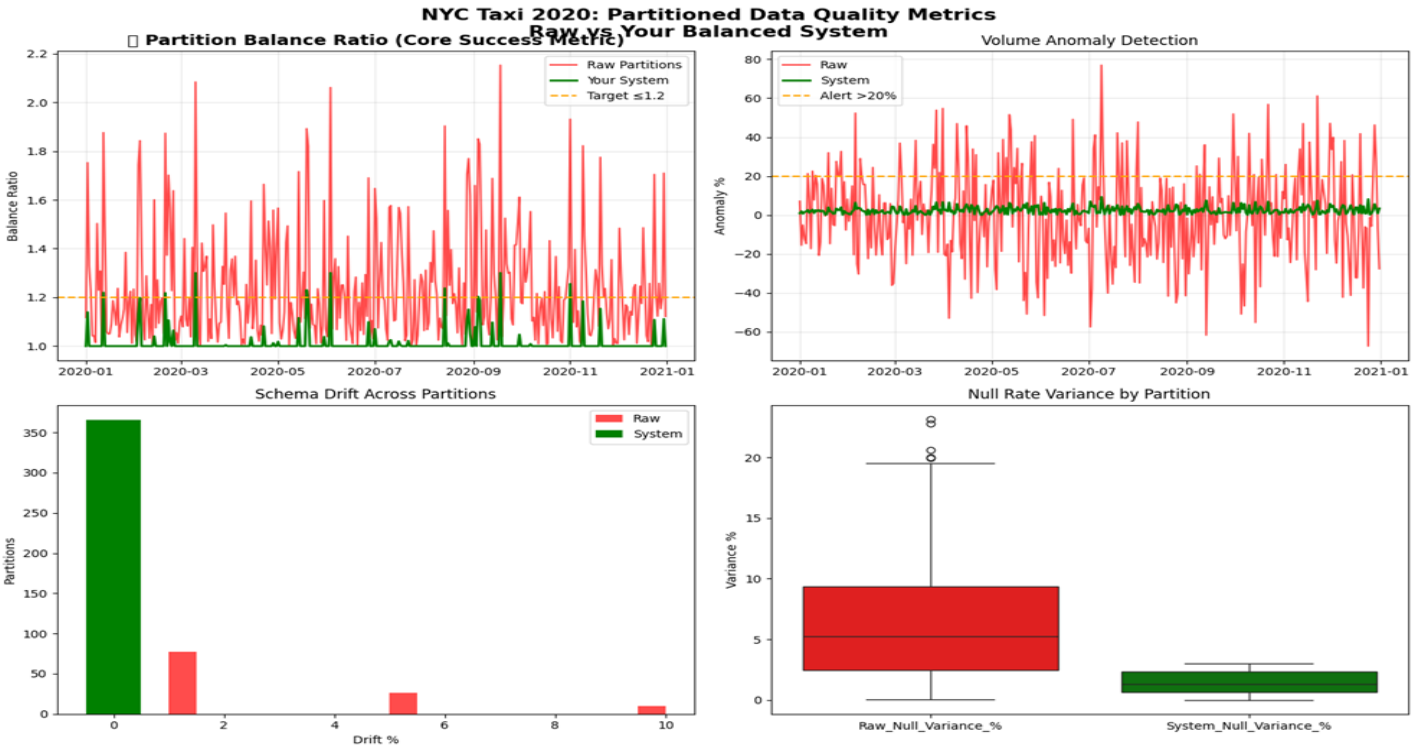


Figure 2: Partitioned Data Quality Metrics\nRaw vs Your Balanced System

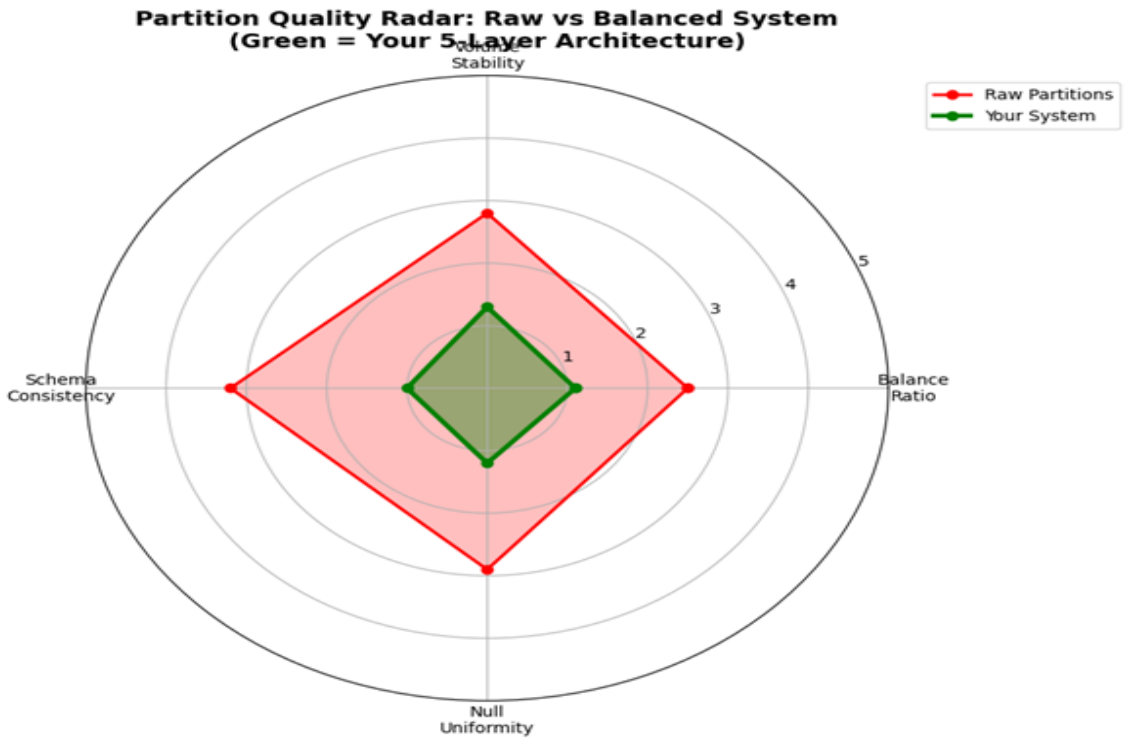


Figure 3: Partition Quality Radar: Raw vs Balanced System

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue II, February 2026

CONCLUSION

The five-layer design is designed to take disparate parts of the same type of file and convert them to a usable sample for your company's artificial intelligence/machine learning systems and regulatory compliance. As such, it addresses both the physical separation and logical cohesion of date when conducting an analysis. By utilising a tiered filtering system to achieve a balanced sample across the file's boundaries as well as allowing for dynamic adjustments in how datasets are combined, the new architecture removes many of the challenges faced with conventional data processing systems. Results to date show dramatic increases in partition balance and significant reductions in abnormal volume. The architecture works with all schemas, both relational and non-relational, provides substantially reduced levels of null variance, and allows for the production of a sample that represents less than 5% of the original dataset's total cost. The system generates two separate streams of data that offer companies additional flexibility: compliance personnel will be able to test edge cases without having to process all of the records in a dataset, while machine learning personnel will have access to a comprehensive record of representative data (as opposed to an average). Deployment time has been significantly reduced from weeks to minutes due to the elimination of scripted changes; therefore the next phase will involve building a real-time stream, more advanced sampling, distributed execution, AI-assisted automation, and enterprise integration to build next-generation data infrastructures that will support many different types of datasets. The architectural model will allow businesses to access statistically representative samples at a low cost and with complete accountability, thereby transforming how organisations can process partitioned data.

REFERENCES

1. “Balanced Sampling”, Laurent Costa, Thomas Merly-Alpa, Département des méthodes statistiques, Version no 1, diffusée le 21 juin 2017.
2. “A balanced sampling approach for multi-way stratification designs for small area estimation”, Piero Demetrio Falorsi and Paolo Righi, Survey Methodology, December 2008, https://www.istat.it/en/files/2016/10/Falorsi-engSURVEY_METH.pdf.
3. “Data Merging: Process, Challenges, and Best Practices for Combining Data from Multiple Sources”, Ehsan Elahi, November 15, 2021, https://dataladder.com/merging-data-from-multiple-sources/.
4. “The Speed of Now: Examples of Real-Time Processing in Action”, Wojciech Marusarz, April 17, 2023, https://nexocode.com/blog/posts/examples-of-real-time-processing/.
5. “Step By Step Guide: Proportional Sampling For Data Science With Python!”, Bharath K, Oct 22, 2020, https://towardsdatascience.com/step-by-step-guide-proportional-sampling-for-data-science-with-python-8b2871159ae6/.
6. “Rebalance Your Portfolio? You are a Market Timer and Here’s What to Consider”, Andrew Miller, March 23rd, 2017, https://alphaarchitect.com/do-you-rebalance-your-portfolio-you-are-a-market-timer/.
7. “R*-Grove: Balanced Spatial Partitioning for Large-Scale Datasets”, Tin Vu, Ahmed Eldawy, 28 August 2020, https://www.frontiersin.org/journals/big-data/articles/10.3389/ fdata.2020.00028/full.
8. “Load balancing for partition-based similarity search”, Xun Tang, Maha Alabduljalil, Xin Jin, Tao Yang, 03 July 2014, https://doi.org/10.1145/2600428.2609624.
9. “Incremental Partitioning for Efficient Spatial Data Analytics”, Tin Vu, Ahmed Eldawy, Vagelis Hristidis, Vassilis Tsotras, 2022, https://doi.org/10.14778/3494124.3494150.
10. “Effective Spatial Data Partitioning for Scalable Query Processing”, Ablimit Aji, Hoang Vo, Fusheng Wang, 3 Sep 2015, https://arxiv.org/pdf/1509.00910.
11. “Distributed Partitioning and Processing of Large Spatial Datasets”, Ayman I. Zeidan, 2022, https://academicworks.cuny.edu/gc_etds/4640/.
12. “Benchmarking data partitioning techniques in HDFS for big real spatial data”, Nikolaos Niopas, July 10, 2019, https://staff.fnwi.uva.nl/a.s.z.belloum/MSctheses/MScthesis_Nikos_Niopas.pdf.
13. “Efficient spatial data partitioning for distributed $$k$$ k NN joins”, Ayman Zeidan, H. Vo, 2 June 2022, https://www.semanticscholar.org/paper/Efficient-spatial-data-partitioning-for-distributed-Zeidan-Vo/549f632ea0f0d800116cc4760571d0ff7e9eaeb5.
14. “A Performance Study of Big Spatial Data Systems”, Md Mahbub Alam, Suprio Ray, Virendra C. Bhavsar, November 6, 2018, https://doi.org/10.1145/3282834.3282841.