International Journal of Advanced Engineering Science and Information Technology (IJAESIT)

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

Distributed Data Engineering Architectures for Smart Retail Inventory Tracking

Dhanaraj Sathiri

Independent Researcher, India

dhanrajsathiri@gmail.com

ABSTRACT: Inventory tracking is crucial for effective retail operations that rely on physical stocks. However, tracking is not simply about maintaining a current state of product stocks but also involves the management of transactions and the analysis of the past. Hence, the data engineering needs for inventory tracking require a distributed architecture that groups functionality into a set of data processing sub-systems connected through a set of shared data flows. Distributed data engineering architectures are examined in terms of all-in-one data streaming and lakehouse designs as well as more modular event-driven designs based on typical data streams for enterprise applications and data lakehouse semantics. Further discussion concerns product and stock data models as well as the elements needed for channel management in a sales system.

Retailers need to take many measures to ensure smooth shopping experiences for their customers. Among these critical actions, being able to manually or automatically monitor the state of physical stocks in a store is important to satisfy customer visits. The concept of inventory tracking is linked to this need; however, inventory tracking is not simply about maintaining the current state of product stocks. Rather, it includes management of sold and purchased stocks, background processes that update the stocks, and supporting transactional data. Being able to evaluate whether a certain product was made available for sale in time and under what conditions during its exposure time is crucial information for taking stock and sales channel decisions. Hence, the data engineering needs for inventory tracking require a distributed architecture that groups functionality into a set of data processing sub-systems connected through a set of shared data flows.

KEYWORDS: Smart retail, inventory tracking, edge computing, Internet of Things, distributed data engineering, data streaming, event-driven architecture, lakehouse, lake architecture.

I. INTRODUCTION

Smart retail uses advanced technologies to create a seamless shopping experience for consumers. Real-time inventory tracking supports many of these technologies by detecting customers' shopping activities from entering the store to completing their purchases. Passive detection techniques based on radio frequency identification (RFID), computer vision, and sensor fusion substantially reduce the burden of inventory management. Data generated by these passive detection systems can be complex, heterogeneous, and produced at a large scale. The architecture for distributed data engineering must be carefully designed to ensure that the supply chain can be supported by data pipelines that combine data streaming with batch processing, handle heterogeneous data types, ensure data quality, manage data consistency, implement storage tiering, and integrate data into a unified model.

Data streaming and event-driven architectures enable support for disseminating detected customer activities to interested parties via a publish-and-subscribe mechanism. A data lakehouse architecture promotes the decoupling of storage and processing for stateful systems such as inventory tracking. A lake architecture augments data lakehouse concepts to enable keeping the latest stock-take and product information in an easily accessible format while using the lakehouse part for long-term storage and archival or analytical purposes. Specific data models expose the semantics of the product, stock, and transaction entities concerning their temporal and spatial dimensions.

Data streaming and event-driven architectures play a crucial role in modern data platforms by enabling real-time communication of detected customer activities to multiple interested systems through a publish-and-subscribe mechanism. In this approach, events such as purchases, product views, or inventory updates are streamed continuously and published to data channels where subscribed services—such as recommendation engines, analytics platforms, or notification systems—can react immediately. Complementing this, a data lakehouse architecture supports the decoupling of storage and processing, allowing stateful systems like inventory tracking to manage large volumes of operational data efficiently. Building further on this concept, a lake architecture enhances the lakehouse by maintaining the most recent stock-take and product information in a readily accessible format for operational use, while the

International Journal of Advanced Engineering Science and Information Technology (IJAESIT)
|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|
DOI: 10.15662/IJAESIT.2022.0506005

lakehouse layer manages long-term storage, historical records, and analytical workloads. To ensure meaningful interpretation of the data, specialized data models define the semantics of product, stock, and transaction entities, incorporating both temporal dimensions (such as time-based changes in inventory or sales) and spatial dimensions (such as store or warehouse locations).

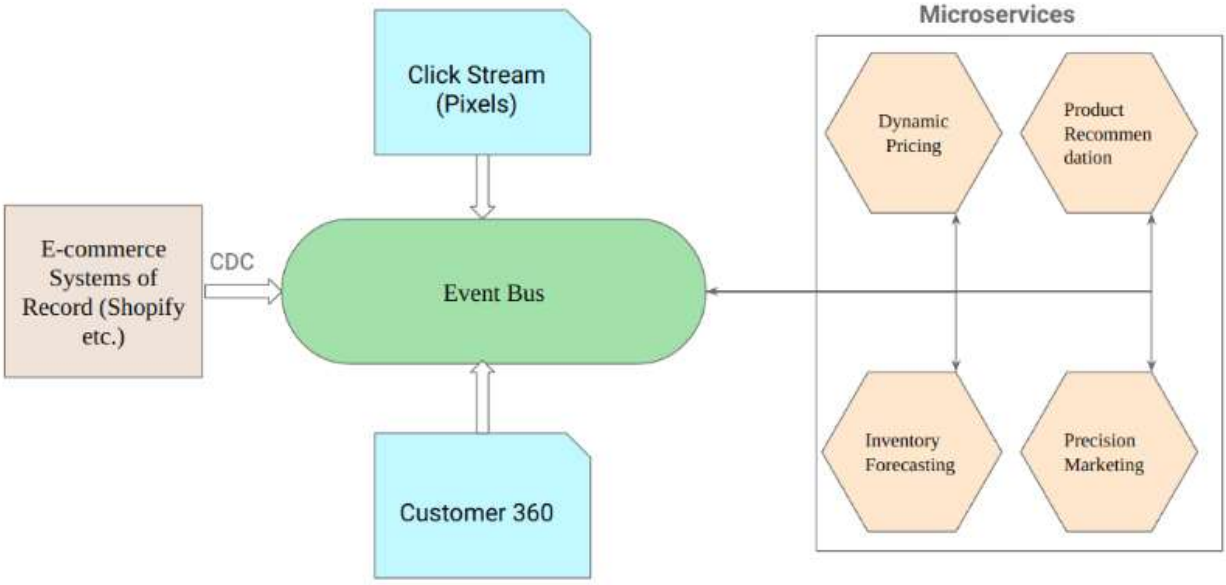


Fig 1: Real Time Data Architecture In Retail

1.1. Background and Significance

Increasingly automated retail stores continuously monitor the presence, availability, and use of products by consumers. Products are equipped with electronic identifiers, such as RFID tags, which are tracked by readers, either fixed at specific locations or carried by customers enabling more granular tracking of usage over time. Reading operations automatically update a central system, which maintains the presence and stock level of products in each store, as well as records sales transactions. These systems generate a continuous stream of events, which can be leveraged by analyzing instant changes of the stock level and by detecting out-of-stock situations.

Such systems operate a data hub that supports multiple analytical and operational business functions in near real-time. These functions include near-real-time stock-out detection enabling automatic replenishment management; detection of stock-outs on frequently purchased items; support of marketing campaigns aiming to boost sales of specific brands; transit monitoring of promotion items misplaced in other non-target stores; and hourly reporting of detailed sales transactions. Business rules capture the resilience of retail business processes and guide the development of dedicated pipelines.

Equation 1: Core inventory balance equation

1. Step 1: Start with stock at the previous observation time t_{k-1}. Denote it by S_{p,l}(t_{k-1}).
2. Step 2: Add all positive movements during the interval (t_{k-1}, t_k], namely inbound replenishments I_{p,l}(t_k) and valid returns R_{p,l}(t_k).
3. Step 3: Subtract all negative movements during the same interval, namely sales/removals O_{p,l}(t_k), damage/shrinkage D_{p,l}(t_k), and any unresolved unknown adjustment U_{p,l}(t_k).
4. Step 4: Combine the terms to obtain the discrete-time balance:

$S_{p,l}(t_k) = S_{p,l}(t_{k-1}) + I_{p,l}(t_k) + R_{p,l}(t_k) - O_{p,l}(t_k) - D_{p,l}(t_k) - U_{p,l}(t_k)$

II. PROBLEM STATEMENT AND REQUIREMENTS

Accurate and up-to-date knowledge about product stocks is essential for a retailer's business success. Unfortunately, retail inventories are often poorly maintained and recorded with significant delay and inaccuracies. A distributed data engineering architecture is proposed for smart retail inventory tracking that captures information about stocks and stock changes as they happen by means of automatic identification technologies. Such data are ingested, stored, and made available in a timely manner supporting a variety of use cases ranging from simple dashboards to AI models for

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|
DOI: 10.15662/IJAESIT.2022.0506005

demand forecasting. Real-time, near real-time, and batch processing architectures are employed, together with state, event, and temporal data models.

Data is captured by automatic identification technologies such as RFID, camera vision, and sensor networks. An RFID installation on a shelf exploits tag reads to identify products being taken from and placed on the shelf. Shelf capacity is enabled by a sensor network that detects when products are near the shelf's end. Additional tag reads from products in the vicinity of the register or scanned by customers contribute to the detection of sales transactions, while a camera vision system is installed behind the cashiers to detect when products are leaving and to generate re-stocking requests. Stock changes are ingested in real time, enriched using event-driven techniques, and stored in a real-time system for dashboarding and other use cases that operate on known or near-actual states. Inventory states are used to answer "what is currently in stock" dashboards and also provide the expected answer for questions such as "what will be in stock in the next hour" if history repeats.

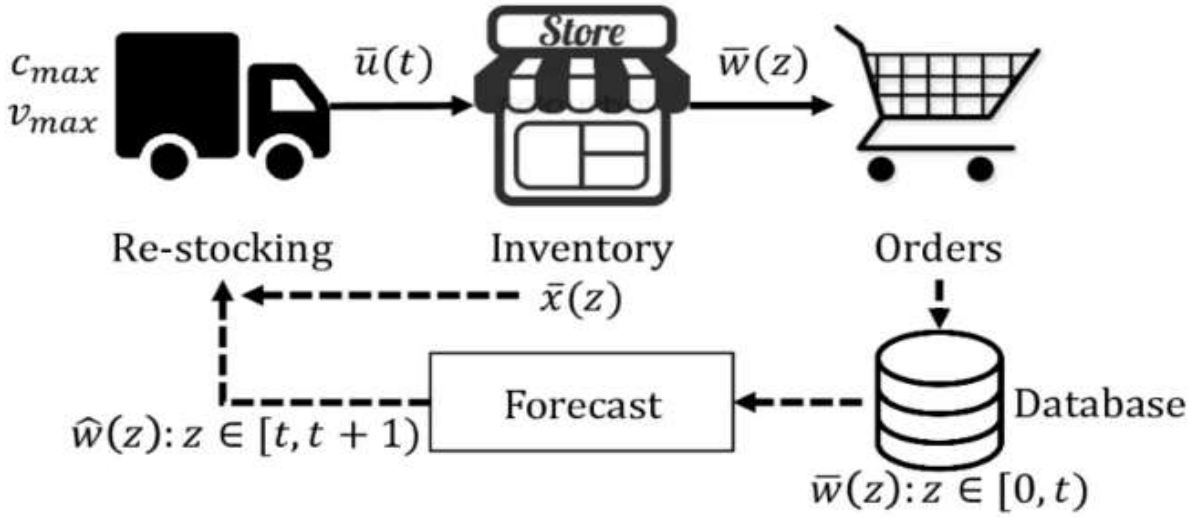


Fig 2: Problem Statement and Requirements of Distributed Data Engineering

2.1. Research design

This research uses an interpretive technology design approach to develop a distributed Data Engineering architecture tailored to enable smart retail inventory tracking. Three architectural qualities—real-time data streaming, unstructured or semi-structured data capture, and data quality management—are considered both singly and in combination and their implications examined through two semi-structured case studies: one exploring food loss in retail and the other focusing on product availability on grocery shelves. For both cases, managed physically distributed, logical cluster architectures combine heterogeneous mutable-data stores with shared-nothing processing. The solutions are deployed via clusters of widely available compute nodes, minimizing operational expenditures. Further, the system responsible for maintaining real-time product-state and -placement information applies a micro-batching technique that captures discrete events to mitigate inconsistencies due to unseen product-state changes. Finally, the architectures of the deployments are documented as precedents for other investigators.

Tracking product stock on supermarket shelves is important for consumer satisfaction and impacts purchasing decisions. Lack of availability on grocery shelves can lead to consumer dissatisfaction, impacting loyalty and long-term revenue. Tracking product stock and availability is essential for consumer satisfaction. A wide variety of items in food retail makes accurate stock monitoring a problem. Important product category for many retailers. The impact of food waste in retail is significant and was estimated at about 41 billion euros for the European retail sector in 2021.

Equation 2: Continuous-time form

Step 1: Suppose stock changes continuously because streaming events arrive at fine time granularity.
Step 2: Replace finite interval changes by rates i_{p,l}(t), r_{p,l}(t), o_{p,l}(t), d_{p,l}(t), and u_{p,l}(t).
Step 3: The net rate of stock change is the sum of positive rates minus the sum of negative rates:
dS_{p,l}(t)/dt = i_{p,l}(t) + r_{p,l}(t) - o_{p,l}(t) - d_{p,l}(t) - u_{p,l}(t)
Step 4: Integrate both sides from time t_0 to t to recover the state equation:
S_{p,l}(t) = S_{p,l}(t_0) + integral from t_0 to t of (i+r-o-d-u) dτ

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|
DOI: 10.15662/IJAESIT.2022.0506005

III. ARCHITECTURAL PARADIGMS FOR DISTRIBUTED DATA ENGINEERING

Data Engineering is a foundational pillar of Data Science that focuses on the acquisition, preparation, and support of data for analysis. Similar to Data Science in the wider sense, Data Engineering combines expertise and technologies from numerous disciplines and areas. Among these, Data Streams, Event-Driven Architecture, Data Lakes and Lakehouses are increasingly becoming of paramount importance for both surrounding developments in other domains and also within Data Engineering itself, moving the focus away from traditional batch-oriented pipeline processes. As a specific example, within the complex ecosystem of smart retail inventory-tracking products and systems, ultra-fast stream ingestion, semantic data integration, and wide-area test-data generation have emerged as the main areas of interest.

The Event-Driven Architecture layer covers everything from the generation of machine-generated data from the S-ETL store for operational processing in U-ETL ingest pipelines, to the subsequent real-time triggering of multiple stacks in Data Science and Business Intelligence. Both Data Lake and Lakehouse concepts unite the two extremes, fast-continual ingestion and ultra-slow reading, into a single, semantically structured entirety that supports both real-time event-driven applications and long-term Data Warehouse-like analysis.

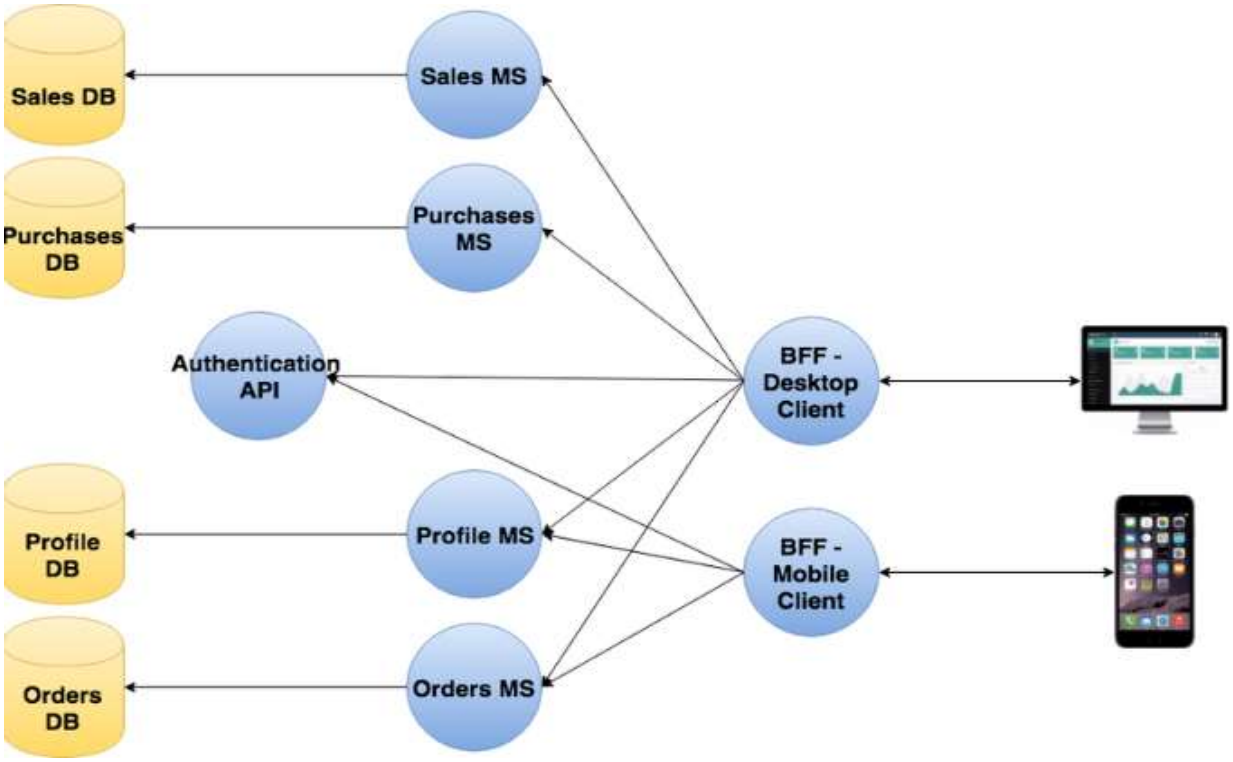


Fig 3: High-level architecture design diagram for Inventory Management

3.1. Data Streaming and Event-Driven Architectures

Smart retail inventory tracking involves supporting applications with strict requirements on the freshness and completeness of inventory data. A possible violation of these requirements can be detected through the modelling of relevant concepts. Distributed data engineering paradigms that are effective in addressing these requirements are discussed.

Modern retail chains frequently adopt a smart retail approach, involving a combination of data-driven services and automations concentrated on customer engagement. Examples of services are chatbots and digital signage that provide visual and textural information on products. Critical for these services is the real-time availability of data describing products and stock levels. Such data are built and maintained by distributed data engineering systems sourced, for example, from inventory transactions and from Internet of Things (IoT) devices that track inventory movements outside of the physical chain, such as self-service checkouts, mobile applications, RFID sensors, or IoT atom boxes. However, these systems do not guarantee that the data are either complete or sufficiently fresh before being accessed by the services. In the case of stock levels, non-temporally consistent data can lead customers to perceive an out-of-stock

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

condition when the stock level is in fact greater than zero. Stale data can substantially degrade the customer experience. Stale information can affect the automatic reordering of products that run low, as well as the sales of products that will expire before being sold.

Equation 3: Event-driven micro-batch update
Step 1: The article mentions micro-batching to reduce inconsistencies from unseen product-state changes.
Step 2: Let a batch B_m contain events e_1, e_2, ..., e_n captured in a small window Δt_m.
Step 3: Let each event contribute a signed stock change Δs(e_j).
Step 4: The batch contribution equals the sum of event deltas:
ΔS(B_m) = sum over e_j in B_m of Δs(e_j)
Step 5: Apply that batch delta to the previous state:
S^{(m)}_{p,l} = S^{(m-1)}_{p,l} + ΔS(B_m)

3.2. Data Lakehouse and Lake Architectures

The lakehouse architectural style supports hybrid transactional—analytical processing (HTAP) workloads that demand low-latency transactions while ensuring data correctness and quality. Lakehouse systems combine the benefits of data warehouses—supporting data quality, data consistency, efficient querying, and support for ACID transactions—with the benefits of data lakes—supporting semi-structured, unstructured, and low-cost storage services at scale.

The lake architecture supports data ingestion and processing in its undifferentiated state as much as possible to minimize transformation and preparation costs. However, unlike the lakehouse, the lake architecture does not support querying and processing with higher quality or correctness. For the inventory tracking system, the lake destination tier is primarily used for historical transactional data for analytics. The lake destination tier does not share the Web interface of the lakehouse; instead, its data is refreshed in batch mode during non-working hours. Lake-sensitive data, for example, product images require storing in the lakehouse for making decisions in the minute-level timescale ETL.

IV. DATA MODELS AND SEMANTICS FOR INVENTORY TRACKING

The ability to achieve and sustain consistency in the real-time state of inventory is fundamentally determined by the semantics and experiments of the product, stock, and transactional entities used in the inventory tracking processes and in the ingestion, storage, and processing of the associated data. Elements of the state of inventory that are tracked explicitly–the entities stored, amount and location of stock in each location category, amount and location of on-the-move stock, locations where inventory transactions are to be processed presently, amount of stock sold, returned, and damaged in each location category during the transactions being processed at present–are naturally stored in the freshest of the data tiers. Any cross-sectional dimensions in the inventory state data and state data quality can change at any point in time. Elements that affect the accuracy of the real-time state of inventory and that cannot be visually detected or tracked presently, e.g., stock thefts and damages, are maintained as unknowns to the real-time tracking processes.

In addition to the semantic foundations that control data quality, the maintenance of quality and the validation of data on-the-fly before its usage in feeds to real-time operations and services (e.g., quality-driven short-term embeddings of the distributions of unknown elements) are critical to ensuring a correct and useful state of inventory at all times. Management of all detected data class and feed-source unknowns is essential for sustaining a consistent real-time state of stock in all categories, for valid estimation of stock in known-to-be-empty service categories (mainly expensive stock), and for the automation of fraud detection. Quality checks at the detailed data source level (e.g., condition checks on images of returned stock) can avoid harmful servicing of automated via more-mature-quality validation strategies. Validation checks at scales larger than those currently thought feasible, including any one of additive, error-controlled summation, and posterior distribution validation checks, can enable correct real-time servicing at desired scale without quality bottlenecks.

International Journal of Advanced Engineering Science and Information Technology (IJAESIT)
|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|
DOI: 10.15662/IJAESIT.2022.0506005

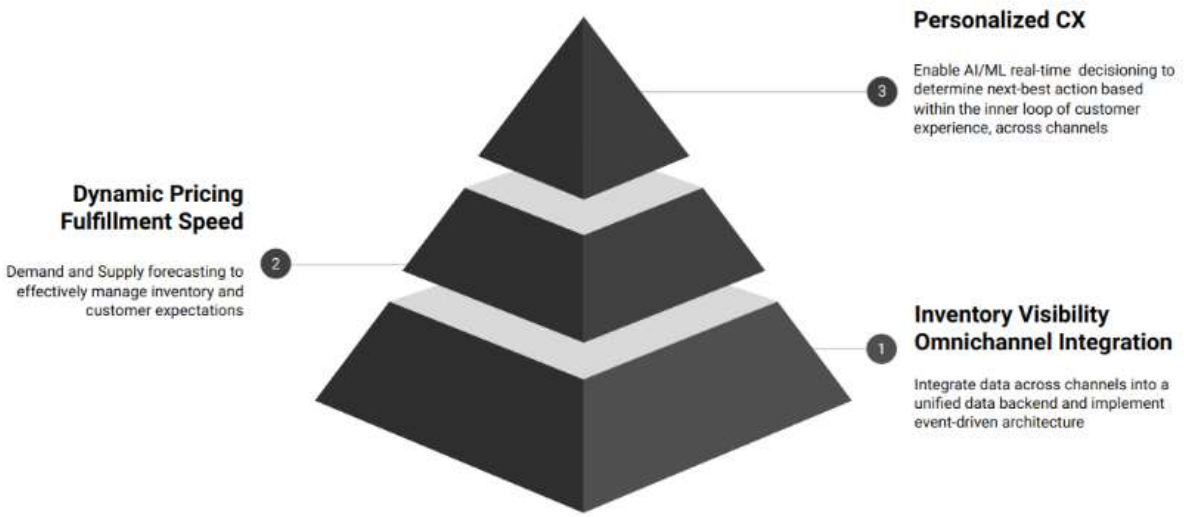


Fig 4: Real time data is important in Retail

4.1. Product, Stock, and Transactional Entities

Referring to product, stock, and transactional entities, an inventory-tracking datacube supporting stakeholder operations and business analytics considers products in a given set with identifiable properties and for which a transaction log exists. The product entity is expanded on with product class and attributes such as name, description, full price, discount percentage, barcode, image, e-commerce information, geographic provenance, and so on; product spatial-temporal provenance may also be specified to support stock replenishment logistics.

Inventory stock and associated transaction logs are generated by incoming and outgoing transactions affecting stock. Stock is represented by a state point table denoting at real time the quantity available at all locations across the product space-time. Inventory entries provide an origin-date product-quantity store-movement description, while outgoing transactions log a user-specified instant-based function transformation over a limited space-time region producing distinct and named—collectible or delivered—stock-group items. A total transaction log tracks all incoming and outgoing transactions at all locations with user reference/debug data; at production origin, it traces produce origin-moving geography and advertisement.

Equation 4: Confidence-weighted inventory under uncertainty

Step 1: Suppose source i reports quantity q_i for a product-location pair.
Step 2: Assign confidence c_i in [0,1] to that observation, where 1 means fully trusted and 0 means unusable.
Step 3: The effective contribution of source i is c_i q_i.
Step 4: Summing trusted contributions over all sources gives the confidence-weighted expected stock:
$\hat{S}_{p,l}(t) = \sum_i c_i(t)\, q_i(t)$
Step 5: If total confidence weights are to be normalized across heterogeneous sources, use the weighted average form:
$\hat{S}_{p,l}(t) = (\sum_i c_i\, q_i) / (\sum_i c_i)$

4.2. Temporal and Spatial Dimensions

Un magasin est un espace dans lequel des produits individuels peuvent être ajoutés et retirés par le biais d'une ou plusieurs transactions. Par conséquent, la structure de données Product doit également contenir une référence à l'espace d'un magasin et la structure de données Time doit contenir un horodatage de transaction. L'ajout d'une dimension temporelle à un magasin permet de mieux s'ajuster à un stock de produits qui ne sont pas inaltérables et qui sont soumis à des variations et à des ruptures de stock. Les données de stock à l'intérieur d'un magasin changent car des produits sont échangés et un magasin se vide ou se remplit au cours de l'année. Un bureau de tabac peut être à court de certains articles pendant trois jours, puis en recevoir de nouveau, ou avoir trop de certains produits, par exemple de la boisson gazeuse pendant la période de Noël, mais ne plus en avoir pendant l'été. Ces échelles de temps doivent également être intégrées à la structure des données. Les ruptures de stock peuvent également survenir pour une durée prolongée, comme pour les rayons qui ne sont pas souvent réassortis, tels que les jouets, les cosmétiques ou le matériel de pêche.
Pour un magasin donné, le produit et l'espace d'horloge représentent un groupe identifiant le stock. Par conséquent, le stock peut être décrit par une structure de données distincte associée à un magasin, à un produit et à un horodatage,

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

avec les quantités nouvelles et supprimées comme propriétés. Un horodatage est nécessaire afin de tenir compte des périodes de rupture de stock.

V. DATA INGESTION, QUALITY, AND CONSISTENCY

Data ingestion pipelines acquire raw messages from sensors or inventory management systems. Automatic connectors handle well-defined streams and queues, as for RFID readers capturing product arrivals at the warehouse gate. Semantic enrichment fills in missing semantics for stores or items, possibly including bar code interpretation. Specialized tools monitor data quality and integrity in key domains. Messages for which entity definitions and semantics are unknown pop up as alerts requiring human attention.

Maintaining a consistent image of the inventory is complicated by asynchronously arriving messages of different types. Hardware arrivals update only the product volume in a store; reader message counts offer a stock snapshot only for the past hour; sales and removals are also important. Yet real-time updates are essential for triggering replenishment orders and alerting shop staff to theft. With very short-lived transactions, sale or removal consistency is relaxed: displayed stock is still the key. Taking these requirements into account, a temporary solution draws stock and sales data into a common NoSQL database dedicated to monitoring, augmented with time dimensions to validate product availability in different stores.

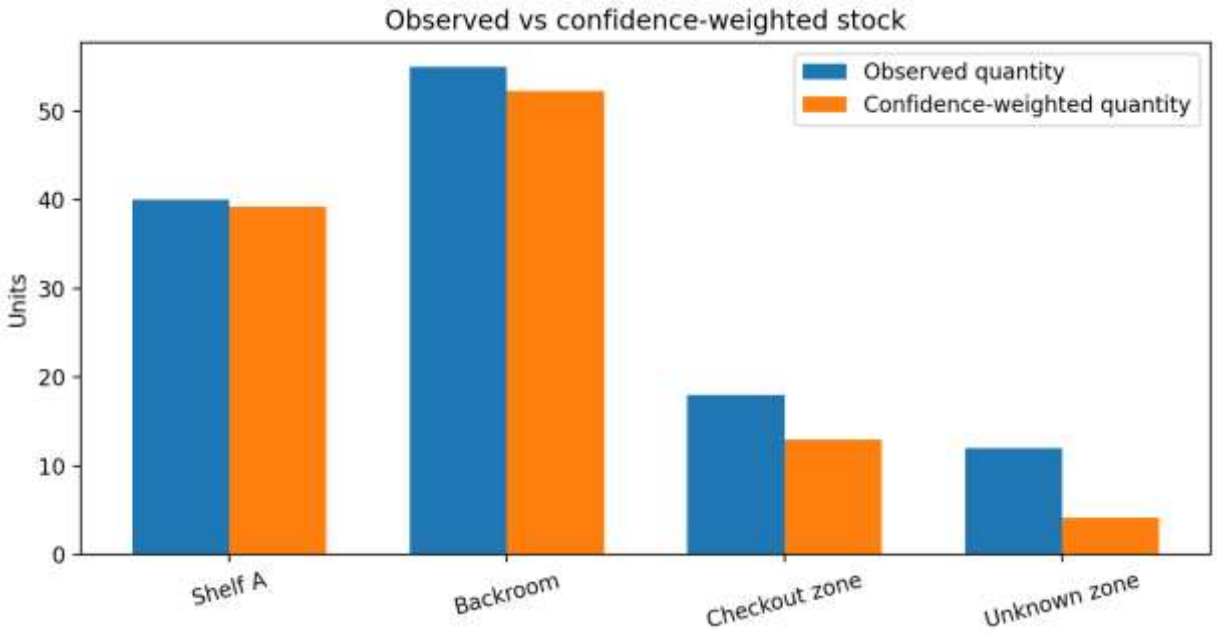
Equation 5: Inventory freshness and latency metrics
Step 1: For event e, define ingestion latency as the difference between processing time and event occurrence time.
$L_e = t_{process}(e) - t_{event}(e)$
Step 2: Over N events, average latency is:
$L_{avg} = (1/N) \sum_{e=1}^{N} L_e$
Step 3: Convert latency into a freshness score that decays as delay grows. A common exponential form is:
$F_e = \exp(-\lambda L_e)$
Step 4: Here $\lambda > 0$ controls how harshly delays are penalized. $F_e = 1$ means perfectly fresh; lower values mean staler data.

5.1. Ingestion Pipelines and Connectors

Data ingestion associates incoming data with appropriate syntactical, structural, and metadata specifications. Syntactical specifications define how input data is represented; structural specifications define how input data is organized; metadata specifications provide context to help validate associated data. Data ingestion solutions incorporate various technology components responsible for collecting and preparing raw data for potential analysis or action. Distributed data engineering architecture designs include ingestion pipelines enabling data collection and dissemination.

Data pipelines extract input from various connectors. Connectors can read from and/or write to data sources provided by other systems via APIs. Connecting software enables interaction with an external system's remote procedure call interface, receiving and/or sending data as problem-specific requests and responses. Stream-based information readily generated and published by a source can also be accessed via message-brokering software products, for the latter effectively facilitating the implementation of data streaming architectures. Data connectors capable of receiving and writing to streams and batches enabled by Apache Kafka, Apache NiFi, Amazon Kinesis, and Google Cloud's Pub/Sub are readily available.

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|
DOI: 10.15662/IJAESIT.2022.0506005



5.2. Data Quality, Validation, and Unknowns

Despite the distributed and decentralised nature of the entire architecture, the implicit assumption remains that the state of the inventory at a point in time is consistent and valid, that an earlier known state is the same as the actually existing stock in the supermarket, and that transactions modify the stock in a deterministic way. Thus, no support for data validation in the ingestion pipelines has been specified. Yet in a real deployment, such an assumption would be unrealistic. Apart from traditional data validation mechanisms, another key aspect associated with the system design would be able to handle that part of the inventory management process and incoming data to which the necessary amount of trust and believe cannot be assigned.

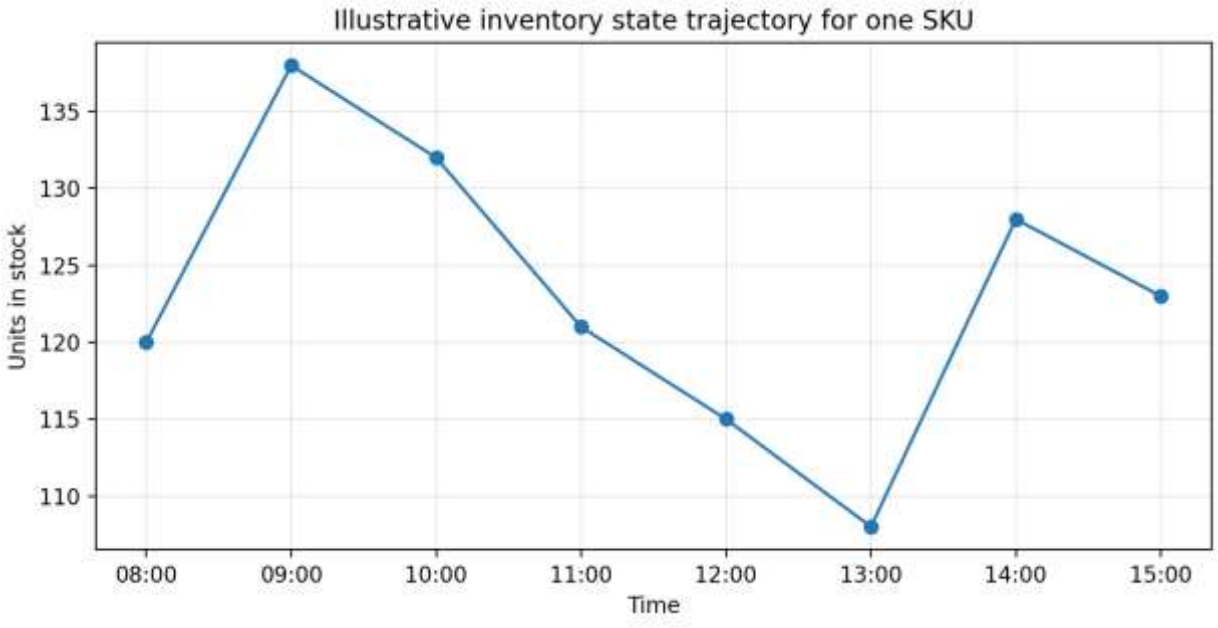
The presence of uncertainty and unknowns in the inventory data is not an uncommon situation, due either to confidence on the existence of some individual or knowledge about its state being lower than the required threshold. Therefore, instead of keeping either the last known real state or just discarding confidence knowledge about the stock, a process based on the Bayesian Formalism can be applied to the stock, discounting each stock quantity according the associated confidence. With additional processing, also knowledge about singleton products being sold out in a given period of time or being refilled could be kept, enabling future usage of such information in the inventory process.

VI. STORAGE TIERING AND METADATA MANAGEMENT

The architectural design delineates two-tiered storage for inventory tracking data. The real-time storage tier of the lakehouse maintains the current stock state in a key-value format, facilitating efficient querying during operations such as smart shelf refills or customer purchases. Each read interface operates on a hot partition holding an up-to-date copy of the stock state, minimizing response times. Read operations leveraging in-memory key-value stores can also be deployed. Yet, the inventory tracking system frequently serves as a data source rather than a center of gravity, supplying real-time data to data warehouse or streaming solutions for customer experience enhancement.

The historical storage tier forms the data lake component. It stores bulk batch updates of the item's state and transaction-based batch updates of incoming and outgoing stock movements—crucial for buyers interested in sales trends. The state management for this component accommodates all data quality issues. Retrofit automates the joining and augmenting of data from source tables to feed the state change table. Ingestion into the state-change table can, indeed, bypass all conformance and transformation steps. Missing data and unknowns are identified, thereby ensuring data-validation steps are undertaken for the real-time copy primarily used by customer-facing applications.

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|
DOI: 10.15662/IJAESIT.2022.0506005



6.1. Real-Time Storage for Inventory State

A distributed data engineering architecture can support inventory-tracking use cases based on a large mixed-city retail product collection—more than 1 million Sellable Product Items belonging to different retailer companies—and including customers, market baskets, transactions, product stock in different products of a grocery store, market baskets of a pharmacy, data of a woman's shoe and handbag.

Although the used technologies support dynamic schema and also fingerprint an entity for flexible handling of unknowns, companies demand a high operational certainty to improve customer experience. Data on Supply Chain changes less rapidly than hourly, so the solution uses a real-time data-store and entry to maintain the live inventory state and a data lake to create a better self-service accessible customer shopping experience, enabling Visit Data Science, addressing the global-trend of Improve Experience with AI. An Event-Driven Architecture uses an Event Source pattern for initial dynamic data ingestion, dynamically storing the inventory product information inside a Lake Storage and maintaining their integrity using a Data-Transformation Service. For out-of-grocery-store and fixed-product Fails forms, a Data-Quality pattern applies automated validation, logic entries in the Data-Quality area of the Supply Chain Configuration Management Tool. Data consistency and integrity are also ensured in standard deployment and handling of unknowns for the transient changes in other Businesses. Popular Deployment cleaning management also provides image sourcing in brand-controlled environments and service on-sell information for audio.

Real-time ingestion and storage of Inventory Supply Chain status enable a next-step Connected-Intelligence use case: providing an Inventory Clothing Management Tool to support company Aprize Clothing. These Supply Chain data are massive and cannot provide spatial competence to Shopping Intelligence use cases.

6.2. Historical Data for Analytics

The architecture of retail inventory tracking systems is typically segregated into distinct storage tiers with varied technology implementations based on different service level requirements. A retained real-time view of all active stock units enables near instantaneous determination of available inventory position during a customer shopping session. While it must be served by high-frequency and performant technology (e.g., NoSQL in memory), tracking systems also tend to accumulate large volumes of historical data that preserve a detailed record of multiple years of transactions spanning every product attribute. This analytical tail of fine-grained transaction history is usually built up as a combination of batch loads and historicized delta synchronizations, consistently with the broader Data Lakehouse paradigm. However, the high ingestion frequency and transactional nature of the data can lead to additional challenges around volatility, skew, and cold-path resource sourcing.

Data architects are increasingly seizing similar opportunities to capitalize on the Data Cloud paradigm, leveraging externalized storage and compute functionality to elastically adapt data volumes to varying time-lengths of reversible cold-path and volatile write workloads. Such capability improves economics, with strategic target-and-park placement

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

of the least expensive and performance-sensitive materialized views enabled by remote storage over the data. Unionized regular views per product and stock unit type allow efficient data discovery and automatic query steering to the most performant resources.

VII. CONCLUSION

Emerging trends in distributed data engineering solutions that unify the analysis of inventory states and transactional data were reviewed. Smart retail environments require inventory routing through shop floors that remain connected in a spatial-temporal stream within a product–stock semantics. The integration of inventory and sales data is crucial for keeping stock levels consistent with temporal patterns in sales data. Processing constraints may arise due to the data-ingestion rate, requiring a schema-less architecture supporting diverse cash register terminals. Recent advances in the streaming–batch continuum and data-lake architecture allow these challenges to be addressed.

The increasing integration of data- engineering solutions and data-science activities toward a Shared-Real-Time layer has the potential to transform the application of analytics in various domains. Although the generation of prediction models continues to be a stimulus for data-science activity, novel data-engineering approaches that support the analytical consumption in real time move to the core of these business intelligence environments. They also assume a foundational role in reinforcement-learning practices that govern control processes.

Equation	Purpose	Link to article concept
S_k = S_{k-1}+I+R-O-D-U	Real-time stock state update	Transactions plus current inventory state
dS/dt = i+r-o-d-u	Continuous stream formulation	Streaming/event-driven architecture
ΔS(B_m)=sum Δs(e)	Micro-batch update	Micro-batching for consistency
Ŝ=sum c_i q_i	Confidence-weighted stock	Bayesian handling of unknowns
L_e=t_process-t_event	Latency measure	Freshness requirement
C=N_received/N_expected	Completeness measure	Data quality monitoring
ROP=μ_τ+zσ_τ	Replenishment trigger	Automatic replenishment
P(stockout)=1-sum e^(-λT)(λT)^k/k!	Risk of running out	Availability forecasting

Table: Compact summary tables

7.1. Emerging Trends

The inventory tracking problem for retail environments is well-suited for experimental deployment of smart retail solutions. During the last few years, several new paradigms and systems for distributed streaming data engineering have appeared. Ready-to-use technologies, including ready-to-use SLAs for various types of products, available connectors or ingestion pipelines, and world sales data feeding a data lake, support the experiments and promote conditions to upgrade a project-proof architecture to a state-of-the-art distributed streaming architecture.

Emerging technologies promise massive adoption in production environments. Data streaming and event-driven architectures enable elastic real-time processing of sudden workloads that are unpredictable in volume and frequency. Lakehouse architectures combine the low-cost storage tiering of data lakes with the performance and schema evolution capabilities of data warehouses. Supported adoption may thus drive customers' frictionless consumption. With minimal specialisation, automated ingestion pipelines can connect supply-side sales stream products to their respective validation procedures and mark resultant violations as unknowns. Unknowns predict products for which sales flow suddenly deviate from common behaviour. Products consuming stale inventories spawn alerts based on temporal and spatial sales concentration shaped by external factors and managed in near real time through controlled false negatives.

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

REFERENCES

[1] Nandan, B. P. (2022). AI-Powered Fault Detection In Semiconductor Fabrication: A Data-Centric Perspective.
[2] Dwaraka Nath Kummari. (2022). AI-Driven Audit Frameworks For Enhancing Compliance In Modern Manufacturing Systems. Migration Letters, 19(S8), 2150–2177. Retrieved from https://migrationletters.com/index.php/ml/article/view/11912
[3] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734–749.
[4] Kothapalli Sondinti, L. R., & Syed, S. (2022). The Impact of Instant Credit Card Issuance and Personalized Financial Solutions on Enhancing Customer Experience in the Digital Banking Era. Universal Journal of Finance and Economics, 1(1), 1223. Retrieved from https://www.scipublications.com/journal/index.php/ujfe/article/view/1223
[5] Arasu, A., & Kaushik, R. (2014). Data cleansing: A context dependent approach. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 135–146.
[6] Phalak, K. Integrating smart transportation systems and urban governance for sustainable mobility: A systematic review. Journal of Urban Management, 13(2), 78–95.
[7] Armbrust, M., Das, T., Davidson, A., Ghodsi, A., Or, A., Rosen, J., Stoica, I., Wendell, P., Xin, R., & Zaharia, M. (2021). Delta Lake: High-performance ACID table storage over cloud object stores. Proceedings of the VLDB Endowment, 13(12), 3411–3424.
[8] Sheelam, G. K., & Nandan, B. P. (2022). Integrating AI And Data Engineering For Intelligent Semiconductor Chip Design And Optimization. Migration Letters, 19, 2178-2207.
[9] Alsharo, M., Alnsour, Y., & Alabdallah, M. (2020). How habit affects continuous use: Evidence from Jordan's national health information system. Informatics for Health and Social Care, 45(1), 43–56. https://doi.org/10.1080/17538157.2018.1540423
[10] Chava, K., Chakilam, C., & Recharla, M. (2021). Machine Learning Models for Early Disease Detection: A Big Data Approach to Personalized Healthcare. International Journal of Engineering and Computer Science, 10(12), 25709–25730. https://doi.org/10.18535/ijecs.v10i12.4678
[11] Chen, X., & Zhang, L. (2022). Real-time big data processing architecture for smart public transportation using Spark and Kafka. Journal of Grid Computing, 20(1), 14–32.
[12] Sriram, H. K. (2022). Advancements in Credit Score Analytics using Deep Learning and Predictive Modeling Techniques. Available at SSRN 5255128.
[13] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. Proceedings of the 2007 SIAM International Conference on Data Mining, 443–448.
[14] Garapati, R. S. (2022). AI-Augmented Virtual Health Assistant: A Web-Based Solution for Personalized Medication Management and Patient Engagement. Available at SSRN 5639650.
[15] Kalisetty, S., & Ganti, V. K. A. T. (2019). Transforming the Retail Landscape: Srinivas's Vision for Integrating Advanced Technologies in Supply Chain Efficiency and Customer Experience. Online Journal of Materials Science, 1, 1254.
[16] Gottimukkala, V. R. R. (2021). Digital Signal Processing Challenges in Financial Messaging Systems: Case Studies in High-Volume SWIFT Flows.
[17] Achar, S., & Devi, M. (2022). A cloud-assisted big data architecture for real-time traffic monitoring and public transit optimization. International Journal of Cloud Computing, 11(3), 245–263.
[18] Dwaraka Nath Kummari. (2022). Fiscal Policy Simulation Using AI And Big Data: Improving Government Financial Planning. Kurdish Studies, 10(2), 934–945. https://doi.org/10.53555/ks.v10i2.3855
[19] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.
[20] Davuluri, P. N. Event-Driven Compliance Systems: Modernizing Financial Crime Detection Without Machine Intelligence.
[21] Das, T., Zhu, A., Li, S., Narayanamurthy, S., & Bhat, P. (2013). Distributed and fault-tolerant streaming computation in Spark. Proceedings of the ACM Symposium on Cloud Computing, 1–12.
[22] Siva Hemanth Kolla. (2022). Knowledge Retrieval Systems for Enterprise Service Environments. International Journal of Intelligent Systems and Applications in Engineering, 10(3s), 495–506. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/8037
[23] Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107–113.
[24] Paleti, S. (2022). Financial Innovation through AI and Data Engineering: Rethinking Risk and Compliance in the Banking Industry. Available at SSRN 5250726.
[25] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. Proceedings of the 21st ACM Symposium on Operating Systems Principles, 205–220.

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

[26] Sriram, H. K., ADUSUPALLI, B., & Malempati, M. (2021). Revolutionizing Risk Assessment and Financial Ecosystems with Smart Automation, Secure Digital Solutions, and Advanced Analytical Frameworks.
[27] Davuluri, P. N. (2020). Improving Data Quality and Lineage in Regulated Financial Data Platforms. Finance and Economics, 1(1), 1-14.
[28] Paleti, S., Singireddy, J., Dodda, A., Burugulla, J. K. R., & Challa, K. (2021). Innovative financial technologies: Strengthening compliance, secure transactions, and intelligent advisory systems through ai-driven automation and scalable data architectures.
[29] Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering, 19(1), 1–16.
[30] Dwaraka Nath Kummari, (2022). Machine Learning Approaches to Real-Time Quality Control in Automotive Assembly Lines. Mathematical Statistician and Engineering Applications, 71(4), 16801–16820. Retrieved from https://philstat.org/index.php/MSEA/article/view/2972
[31] Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). "Counting your customers" the easy way: An alternative to the Pareto/NBD model. Marketing Science, 24(2), 275–284.
[32] Inala, R. (2022). Engineering Data Products for Investment Analytics: The Role of Product Master Data and Scalable Big Data Solutions. International Journal of Scientific Research and Modern Technology, 155-171.
[33] Davuluri, P. N. (2020). Improving Data Quality and Lineage in Regulated Financial Data Platforms. Finance and Economics, 1(1), 1-14.
[34] Meda, R. Enabling Sustainable Manufacturing Through AI-Optimized Supply Chains.
[35] Ghemawat, S., Gobioff, H., & Leung, S. T. (2003). The Google file system. Proceedings of the 19th ACM Symposium on Operating Systems Principles, 29–43.
[36] Inala, R. Advancing Group Insurance Solutions Through Ai-Enhanced Technology Architectures And Big Data Insights.
[37] Yandamuri, U. S. (2021). A Comparative Study of Traditional Reporting Systems versus Real-Time Analytics Dashboards in Enterprise Operations. Universal Journal of Business and Management, 1(1), 1–13. Retrieved from https://www.scipublications.com/journal/index.php/ujbm/article/view/1357
[38] Gottimukkala, V. R. R. (2022). Licensing Innovation in the Financial Messaging Ecosystem: Business Models and Global Compliance Impact. International Journal of Scientific Research and Modern Technology, 1(12), 177-186.
[39] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.
[40] Vadisetty, R., Polamarasetti, A., Guntupalli, R., Raghunath, V., Jyothi, V. K., & Kudithipudi, K. (2022). AI-Driven Cybersecurity: Enhancing Cloud Security with Machine Learning and AI Agents. Sateesh kumar and Raghunath, Vedaprada and Jyothi, Vinaya Kumar and Kudithipudi, Karthik, AI-Driven Cybersecurity: Enhancing Cloud Security with Machine Learning and AI Agents (February 07, 2022).
[41] Kalisetty, S., Vankayalapati, R. K., Reddy, L., Sondinti, K., & Valiki, S. (2022). AI-Native Cloud Platforms: Redefining Scalability and Flexibility in Artificial Intelligence Workflows. Linguistic and Philosophical Investigations, 21(1), 1-15.
[42] Garapati, R. S. (2022). Web-Centric Cloud Framework for Real-Time Monitoring and Risk Prediction in Clinical Trials Using Machine Learning. Current Research in Public Health, 2, 1346.
[43] Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. Proceedings of the 2008 IEEE International Conference on Data Mining, 263–272.
[44] Amistapuram, K. (2022). Fraud Detection and Risk Modeling in Insurance: Early Adoption of Machine Learning in Claims Processing. Available at SSRN 5741982.
[45] Davuluri, P. S. L. N. (2021). Event-Driven Compliance Systems: Modernizing Financial Crime Detection Without Machine Intelligence. Journal of International Crisis and Risk Communication Research, 339–354. https://doi.org/10.63278/jicrcr.vi.3636
[46] Meda, R. (2022). Integrating Edge AI in Smart Factories: A Case Study from the Paint Manufacturing Industry. International Journal of Science and Research (IJSR), 1473-1489.
[47] Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (2014). Big data and its technical challenges. Communications of the ACM, 57(7), 86–94.
[48] Segireddy, A. R. (2020). Cloud Migration Strategies for High-Volume Financial Messaging Systems.
[49] Khatri, V., & Brown, C. V. (2010). Designing data governance. Communications of the ACM, 53(1), 148–152.
[50] Amistapuram, K. (2021). Digital Transformation in Insurance: Migrating Enterprise Policy Systems to .NET Core. Universal Journal of Computer Sciences and Communications, 1(1), 1–17.
[51] Kleppmann, M. (2017). Designing data-intensive applications. O'Reilly Media.
[52] Nagabhyru, K. C. (2022). Bridging Traditional ETL Pipelines with AI Enhanced Data Workflows: Foundations of Intelligent Automation in Data Engineering. Available at SSRN 5505199.
[53] Lahiri, M., & Venkatasubramanian, S. (2013). Robust record linkage. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 101–112.

|ISSN 2349-3216| Volume 5, Issue 6, November-December 2022| Bimonthly, Peer Reviewed and Scholarly Indexed Journal|

DOI: 10.15662/IJAESIT.2022.0506005

[54] Singireddy, J. (2022). Leveraging Artificial Intelligence and Machine Learning for Enhancing Automated Financial Advisory Systems: A Study on AIDriven Personalized Financial Planning and Credit Monitoring. Mathematical Statistician and Engineering Applications, 71 (4), 16711–16728.
[55] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets (2nd ed.). Cambridge University Press.
[56] Adusupalli, B., Pandiri, L., & Singireddy, S. (2019). DevOps Enablement in Legacy Insurance Infrastructure for Agile Policy and Claims Deployment. risk, 7(12).
[57] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 76–80.
[58] Meda, R. (2021). Digital Infrastructure for Predictive Inventory Management in Retail Using Machine Learning. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), DOI, 10.
[59] Lin, J., Kolcz, A., & Szymanski, B. K. (2012). Large-scale machine learning at Twitter. Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 793–804.
[60] Sheelam, G. K. Power-Efficient Semiconductors for AI at the Edge: Enabling Scalable Intelligence in Wireless Systems. International Journal of Innovative Research in Electrical, Elec-tronics, Instrumentation and Control Engineering (IJIREEICE), DOI, 10.
[61] Bulatova, O. Using big data in smart cities transportation systems. E3S Web of Conferences, 371, 06009.
[62] Vadisetty, R., Polamarasetti, A., Guntupalli, R., Rongali, S. K., Raghunath, V., Jyothi, V. K., & Kudithipudi, K. (2021). Legal and Ethical Considerations for Hosting GenAI on the Cloud. International Journal of AI, BigData, Computational and Management Studies, 2(2), 28-34.
[63] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Proceedings of the International Conference on Learning Representations, 1–12.
[64] Ramesh Inala. (2022). Cross-Domain MDM Integration Using AI-Driven Data Governance: A Case Study In Financial Technology Architecture. Migration Letters, 19(2), 280–304. Retrieved from https://migrationletters.com/index.php/ml/article/view/11982
[65] Challa, K. (2021). Cloud Native Architecture for Scalable Fintech Applications with Real Time Payments. International Journal Of Engineering And Computer Science, 10(12).
[66] Aitha, A. R. (2021). Optimizing Data Warehousing for Large Scale Policy Management Using Advanced ETL Frameworks.
[67] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 1–7.
[68] Adusupalli, B., Singireddy, S., Sriram, H. K., Kaulwar, P. K., & Malempati, M. (2021). Revolutionizing Risk Assessment and Financial Ecosystems with Smart Automation, Secure Digital Solutions, and Advanced Analytical Frameworks. Universal Journal of Finance and Economics, 1(1), 101-122.
[69] Zaharia, M., Das, T., Li, H., Shenker, S., & Stoica, I. (2012). Discretized streams: Fault-tolerant streaming computation at scale. Proceedings of the 24th ACM Symposium on Operating Systems Principles, 423–438.
[70] Segireddy, A. R. (2021). Containerization and Microservices in Payment Systems: A Study of Kubernetes and Docker in Financial Applications. Universal Journal of Business and Management, 1(1), 1–17.
[71] Zhai, C., & Massung, S. (2016). Text data management and analysis: A practical introduction to information retrieval and text mining. ACM & Morgan Claypool.
[72] Uday Surendra Yandamuri. (2022). Cloud-Based Data Integration Architectures for Scalable Enterprise Analytics. International Journal of Intelligent Systems and Applications in Engineering, 10(3s), 472–483. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/8005.
[73] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135–146.
[74] Keerthi Amistapuram, "Energy-Efficient System Design for High-Volume Insurance Applications in Cloud-Native Environments," International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering (IJIREEICE), DOI 10.17148/IJIREEICE.2020.81209
[75] Goutham Kumar Sheelam. (2022). Reconfigurable Semiconductor Architectures For AI-Enhanced Wireless Communication Networks. Kurdish Studies, 10(2), 1027–1040. https://doi.org/10.53555/ks.v10i2.3867
[76] Batarseh, F. A., & Yang, R. (2019). Federal data science: Transforming government and society. Academic Press.
[77] Kolla, S. K. (2021). Architectural Frameworks for Large-Scale Electronic Health Record Data Platforms. Current Research in Public Health, 1(1), 1–19. Retrieved from https://www.scipublications.com/journal/index.php/crph/article/view/1372
[78] Ma, Z., & Wu, J. (2022). Hybrid cloud-fog computing architecture for big data analytics in public bus transit systems. Applied Sciences, 12(5), 2341.
[79] Kolla, S. H. (2021). Rule-Based Automation for IT Service Management Workflows. Online Journal of Engineering Sciences, 1(1), 1–14. Retrieved from https://www.scipublications.com/journal/index.php/ojes/article/view/1360

[80] Pandiri, L., Singireddy, S., & Adusupalli, B. (2020). Digital Transformation of Underwriting Processes through Automation and Data Integration. Global Research Development (GRD) ISSN, 2455-5703.
[81] Abedjan, Z., Golab, L., & Naumann, F. (2016). Profiling relational data: A survey. The VLDB Journal, 24(4), 557–581.
[82] Yandamuri, U. S. (2022). Big Data Pipelines for Cross-Domain Decision Support: A Cloud-Centric Approach. International Journal of Scientific Research and Modern Technology, 1(12), 227–237. https://doi.org/10.38124/ijsrmt.v1i12.1111
[83] Challa, K. (2022). The Future of Cashless Economies Through Big Data Analytics in Payment Systems. International Journal of Scientific Research and Modern Technology, 60-70.
[84] Joseph, A. L., Stringer, E., Borycki, E. M., & Kushniruk, A. W. (2022). Evaluative frameworks and models for health information systems (HIS) and health information technologies (HIT). Studies in Health Technology and Informatics, 289, 280–285. https://doi.org/10.3233/shti210914
[85] Baesens, B., Van Vlasselaer, V., & Verbeke, W. (2021). Fraud analytics using descriptive, predictive, and social network techniques: A guide to data science for fraud detection (2nd ed.). Wiley.
[86] Avinash Reddy Segireddy. (2022). Terraform and Ansible in Building Resilient Cloud-Native Payment Architectures. International Journal of Intelligent Systems and Applications in Engineering, 10(3s), 444–455. Retrieved from https://www.ijisae.org/index.php/IJISAE/article/view/7905
[87] Guo, H., Huang, R., & Xu, Z. The design of intelligent highway transportation system in smart city based on the internet of things. Scientific Reports, 14, Article 79903. https://doi.org/10.1038/s41598-024-79903-0.
[88] Avinash Reddy Aitha. (2022). Deep Neural Networks for Property Risk Prediction Leveraging Aerial and Satellite Imaging. International Journal of Communication Networks and Information Security (IJCNIS), 14(3), 1308–1318. Retrieved from https://www.ijcnis.org/index.php/ijcnis/article/view/8609.
[89] Gottimukkala, V. R. R. (2020). Energy-Efficient Design Patterns for Large-Scale Banking Applications Deployed on AWS Cloud. power, 9(12).
[90] Liu, Y., & Ke, L. Cloud-assisted Internet of Things intelligent transportation systems and traffic control in smart cities. IEEE Internet of Things Journal, 10(4), 3120–3135.
[91] Aitha, A. R. (2022). Cloud Native ETL Pipelines for Real Time Claims Processing in Large Scale Insurers. Available at SSRN 5532601.
[92] Aljabre, A. (2019). Cloud computing security in healthcare. Journal of King Saud University – Computer and Information Sciences, 31(1), 10–18.
[93] Gadi, A. L. The Role of Digital Twins in Automotive R&D for Rapid Prototyping and System Integration.
[94] Weber, K., Otto, B., & Österle, H. (2009). One Size Does Not Fit All—A Contingency Approach to Data Governance. Journal of Data and Information Quality, 1(1), 1–27.
[95] Goutham Kumar Sheelam, "Semiconductor Innovation for Edge AI: Enabling Ultra-Low Latency in Next-Gen Wireless Networks," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), DOI: 10.17148/IJARCCE.2022.111258
[96] Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. M. (2013). Data Management in Cloud Environments: NoSQL and NewSQL Data Stores. Journal of Cloud Computing, 2(22), 1–24.
[97] Vadisetty, R., Polamarasetti, A., Guntupalli, R., Raghunath, V., Jyothi, V. K., & Kudithipudi, K. (2021). Privacy-Preserving Gen AI in Multi-Tenant Cloud Environments. Sateesh kumar and Raghunath, Vedaprada and Jyothi, Vinaya Kumar and Kudithipudi, Karthik, Privacy-Preserving Gen AI in Multi-Tenant Cloud Environments (January 20, 2021).
[98] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A View of Cloud Computing. Communications of the ACM, 53(4), 50–58.
[99] Davuluri, P. N. (2020). Event-Driven Architectures for Real-Time Regulatory Monitoring in Global Banking.
[100] Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The Rise of Big Data on Cloud Computing: Review and Open Research Issues. Information Systems, 47, 98–115.
[101] Dodda, A., Lakkarasu, P., Singireddy, J., Challa, K., & Pamisetty, V. (2022). Optimizing Digital Finance and Regulatory Systems Through Intelligent Automation. Secure Data Architectures, and Advanced Analytical Technologies.
[102] Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. Mobile Networks and Applications, 19(2), 171–209.
[103] Kannan, S. (2021). Advanced Computational Technologies in Vehicle Production, Digital Connectivity, and Sustainable Transportation: Innovations in Intelligent Systems, Eco-Friendly Manufacturing, and Financial Optimization. Universal Journal of Finance and Economics.
[104] Janssen, M., & Kuk, G. (2016). Big and Open Linked Data in Government: A Challenge to Transparency and Privacy? Government Information Quarterly, 33(2), 363–368.
[105] Annapareddy, V. N., Preethish Nandan, B., Kommaragiri, V. B., Gadi, A. L., & Kalisetty, S. (2022). Emerging Technologies in Smart Computing, Sustainable Energy, and Next-Generation Mobility: Enhancing Digital Infrastructure, Secure Networks, and Intelligent Manufacturing.