WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Review of Data Engineering Frameworks (Trino and Kubernetes) for Implementing Generative AI in Financial Risk

Satyadhar Joshi

BoFA, Jersey City, NJ

A B S T R A C T

This paper is discusses Data Engineering Full Stack Framework for using Generative AI (GenAI) in financial risk management. We have discusses challenges, proposals and application of various data tools directly applicable for implementation of Gen AI in Financial risk. We have discussed issues and comparison of tools for data engineering pertinent to Generative AI (GenAI) is transforming financial risk management by leveraging Gen AI models to generate synthetic data for stress testing, anomaly detection, and risk modeling. This paper explores full-stack big data frameworks, including Apache Spark, Trino, Kubernetes, AWS SageMaker, and Databricks, to deploy GenAI models at scale. This paper furthur provides a literature review of AI and Generative AI applications in Trino, discussing query performance, optimization techniques, and AI-driven analytics. Additionally, we explore the bidirectional synergy between Trino and Generative AI—how AI enhances Trino's capabilities and how Trino facilitates advanced AI workloads. Key contributions include insights from recent research on Trino's scalability, AI-powered query execution, and the trade-offs between generative models and computational efficiency. Deployment strategies leveraging AWS, Kubernetes, and Apache Spark for scalable risk computation are also discussed. Generative AI (GenAI) is transforming financial risk management by leveraging deep learning models to generate synthetic data for stress testing, anomaly detection, and risk modeling. This paper also reviews the latest developments in Azure-based AI agents and Kubernetes orchestration for AI/ML workloads. We categorize recent advancements, including agentic workflows, AI/ML orchestration, Kubernetes integration, and AI-driven DevOps.

Keywords: GenAI Full-Stack Framework, Date Engineering for GenAI, Trino, Kubernetes

1. Introduction

Financial institutions face challenges in risk modeling due to market volatility and incomplete data. Generative AI can enhance risk assessment by producing synthetic financial scenarios. This paper proposes a scalable full-stack architecture integrating big data technologies with GenAI models for real-time inference. This paper proposes a scalable full-stack architecture integrating big data technologies with GenAI models for real-time inference and cloud-based scalability using different Data Engineering tools. Trino, formerly known as PrestoSQL, is a distributed query engine that enables high-performance SQL analytics across multiple data sources. The integration of AI and Generative AI into Trino has expanded its capabilities, particularly in natural language query processing, automated query optimization, and predictive analytics. However, these enhancements bring forth performance challenges such as increased computational overhead, model inference latency, and resource allocation issues. The rapid advancement of AI and Kubernetes has reshaped modern cloud computing environments. With AI agents automating DevOps workflows, organizations can now leverage Kubernetes' robust orchestration capabilities to scale AI/ML workloads efficiently.

2. Build up from Past Work

In our earlier work we have explored the use of Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformer-based models for financial risk prediction. Synthetic data generation improves stress testing, scenario analysis, and anomaly detection. This section discusses our prior work related to the application of Generative AI (GenAI) in financial risk management.

Joshi explores the synergy between GenAI and Big Data for financial risk assessment in [1]. The potential of GenAI to improve the resilience of the US financial and regulatory framework is examined in [2], where he investigates the use of advanced models such as ChatGPT-4 and Google Gemini for generating relevant queries for existing financial risk models. A comprehensive review of GenAI models tailored for financial risk management is presented in [3], covering aspects like fine-tuning GPT models with proprietary data and employing zero-shot classification techniques. With regards to time-series and alternative data (macro, ESG, sentiment), we have earlier proposed a GAN-based data generation model for financial. In [4], Joshi delves into the application of GenAI, specifically focusing on Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), to enhance established structured finance risk models like the Leland-Toft and Box-Cox models. His book, Agentic Gen AI For Financial Risk Management [5],

* Corresponding author. Tel.: +1-929-356-5046
E-mail address: satyadhar.joshi@gmail.com

provides a more extensive exploration of the topic, offering a broader perspective on the integration of GenAI in financial risk management. Joshi further investigates the advancement of financial risk modeling by enhancing the Vasicek framework with agentic generative AI in [6]. A dedicated literature review focusing on GenAI agents within financial applications, covering both models and their implementations, is presented in [7]. The crucial role of prompt engineering in improving financial market integrity and risk management using Large Language Models (LLMs) like ChatGPT-4 and Google Gemini is examined in [8]. Joshi also addresses the essential aspects of data engineering and the role of data lakes in effectively implementing GenAI for financial risk management in [9]. Furthermore, he explores the broader implications of agentic generative AI on the future of the US workforce, particularly in terms of innovation and national competitiveness, in [10]. These publications collectively demonstrate Joshi's extensive contributions to the field of GenAI in finance, covering a wide range of topics from model development and implementation to broader societal impacts. Joshi's GitHub repository [11] serves as a central hub for his projects and code related to this field. In our earlier work we have shown how generative AI models and large language models (LLMs) complement each other in financial risk analysis by enhancing data synthesis, interpretation, and automation. While Generative AI, including GANs and VAEs, focuses on creating realistic synthetic financial datasets, LLMs like GPT-4 and BERT provide interpretability, automated reporting, and domain-specific risk assessment.

3. Relation Between Generative AI and LLM-Based Models in Big Data Frameworks

Key intersections between GenAI and LLMs in financial big data frameworks include includes:

• Hybrid AI Workflows(from our last work): Combining GenAI and LLMs in a unified pipeline allows financial institutions to generate realistic stress scenarios, analyze large volumes of structured and unstructured data, and provide explainable AI-driven risk predictions.

• System Properties Comparison: The integration of GenAI and LLMs within big data architectures involving Teradata, Trino, and Vertica ensures optimized storage, retrieval, and computation for scalable financial risk management.

This synergy between generative and LLM-based models significantly enhances risk evaluation frameworks by improving robustness, interpretability, and automation in financial analytics.

3.1 Generative AI for Market Risk: Variational Autoencoders (VAEs)

VAEs model financial returns distribution via a latent variable z:

$$p(x) = \int p(x \vee z)p(z)dz$$

where $p(x \vee z)$ is a neural network decoder and $p(z)$ is a Gaussian prior.

Python Code:

```
import torch
from torch import nn
 class VAE(nn.Module):
def __init__(self, input_dim, latent_dim):
super(VAE, self).__init__()
self.encoder = nn.Sequential(nn.Linear(input_dim, 128), nn.ReLU(),
nn.Linear(128, latent_dim*2))
self.decoder = nn.Sequential(nn.Linear(latent_dim, 128), nn.ReLU(),
nn.Linear(128, input_dim))
 def forward(self, x):
mu, log_var = self.encoder(x).chunk(2, dim=-1)
      std = torch.exp(0.5 * log_var)
      z = mu + std * torch.randn_like(std)
      return self.decoder(z), mu, log_var
      ae = VAE(input_dim=10, latent_dim=2)
```

3.2 Generative Adversarial Networks (GANs)

GANs generate synthetic financial returns using:

$$\min_{G}\max_{D} E\big[logD(x)\big) + E\left[log\left(1 - D\big(G(z)\big)\right)\right)$$

where G is the generator and D is the discriminator.

Python Code:

```
class Generator(nn.Module):
    def __init__(self, input_dim):
    super().__init__()
    self.model = nn.Sequential(nn.Linear(input_dim, 128), nn.ReLU(),
    nn.Linear(128, 1))
        def forward(self, z):
    return self.model(z)
    G = Generator(input_dim=10)
```

3.3 Transformer-Based Market Risk Prediction

Transformers model sequential dependencies in time series using **self-attention**:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Python Code:

```
import torch.nn.functional as F
    def scaled_dot_product_attention(Q, K, V):
    scores = torch.matmul(Q, K.transpose(-2, -1)) / np.sqrt(K.shape[-1])
    attn = F.softmax(scores, dim=-1)
    return torch.matmul(attn, V)
```

4. Trino and GenAI

Formerly known as PrestoSQL, Trino has emerged as a powerful distributed SQL query engine designed for low-latency data processing across various data sources. Literature underscores its effectiveness in providing real-time analytics, particularly when paired with modern data lake architectures. Key advancements in Trino include improved query optimization techniques, support for ANSI SQL standards, and enhanced compatibility with cloud-native storage solutions. Recent case studies have demonstrated its successful deployment in industries such as finance, healthcare, and e-commerce for interactive data analysis and reporting.

Trino is an open-source distributed SQL query engine optimized for interactive analytics on large datasets. Originally developed at Facebook, Trino has evolved into a robust solution for querying diverse data sources, including Hadoop, RDBMSs, and NoSQL systems [12]. With the rise of generative AI, Trino has also been integrated into natural language interfaces for data exploration [13]. This section provides a discssion of Trino's architecture, performance, and use cases. We compare Trino and PrestoDB, its predecessor, and present quantitative data to highlight its advantages. Additionally, we include pseudocode for query execution and discuss its integration with AI-driven analytics.

4.1 Bidirectional Synergy: Generative AI and Trino

The interaction between Generative AI and Trino is twofold:

Generative AI Enhancing Trino: AI models improve Trino's functionality through query optimization, anomaly detection, and natural language interfaces [13].

Trino Facilitating AI Workloads: Trino provides a scalable, distributed computing environment for training and deploying AI models, enabling efficient data retrieval for AI-driven applications [14].

This bidirectional synergy enables new innovations in AI and data processing:

Enhanced AI Model Training: Trino's distributed processing speeds up AI model training by efficiently handling large datasets [15].

AI-Driven Query Execution: Generative AI assists in predicting query patterns and optimizing execution plans [16].

Scalable Data Pipelines: AI models leverage Trino's federated query capabilities to access diverse data sources efficiently [17].

This mutual reinforcement enables Trino to become more AI-driven while also serving as an infrastructure backbone for AI research and deployment.

4.2 Generative AI in Trino: Applications and Challenges

Generative AI, particularly large language models (LLMs), has been explored for enhancing Trino's functionality. Applications include:

- **Natural Language Interfaces:** AI-driven interfaces enable users to interact with Trino using natural language queries [13].
- **Automated Query Optimization:** Machine learning models suggest efficient query execution plans [14].
- **Anomaly Detection:** AI models identify query performance anomalies and suggest corrective actions [18].

However, challenges persist:

- **Inference Latency:** Generative AI models introduce delays in query execution [19].
- **Scalability Concerns:** AI-enhanced Trino queries may not scale efficiently across large datasets [20].
- **Cost Overheads:** Running AI models alongside Trino requires substantial computational resources [15].

4.3 Literature Review on Hive and Trino ODBC Integration

The integration of Hive and Trino through ODBC connections has been the subject of numerous studies and practical implementations, primarily driven by the need for scalable and efficient data processing solutions in big data environments.

Recent studies have provided quantitative insights into Trino's performance when integrated with AI-based query optimization. For example:

- A study by [12] demonstrated that AI-enhanced query execution improved response time by up to 30% while increasing computational overhead by 15%.
- [17] reported that AI-driven query optimizations reduced memory consumption by 20% in distributed environments.

4.4 System Properties Comparison: Teradata vs. Trino vs. Vertica

Big data architectures supporting Generative AI in financial risk management require optimized database solutions. This section compares three popular distributed query engines—Teradata, Trino, and Vertica—based on their applicability to GenAI-driven workloads.

Teradata: Known for its high-performance analytics capabilities, Teradata offers robust SQL-based query optimizations and parallel processing. It is well-suited for large-scale structured financial data but may require extensive tuning for unstructured AI-driven workloads.

Trino: An open-source distributed SQL query engine optimized for real-time analytics. Trino excels in federated querying across heterogeneous data sources, making it highly adaptable for GenAI models that require integration with multiple financial datasets, including streaming and historical data.

Vertica: Designed for high-speed analytical processing, Vertica provides advanced machine learning capabilities and columnar storage. It is particularly efficient for AI-powered risk assessment by supporting scalable in-database analytics and predictive modeling directly within the database.

From a GenAI perspective, Trino's flexibility in querying diverse data sources makes it advantageous for real-time inference, while Vertica's built-in analytics provide a strong foundation for AI-enhanced risk assessments. Teradata remains a preferred choice for structured financial data processing but may require additional layers to fully support Gen AI workflows.

5. Discussions on Data Engineering Tools for Generative AI powered Financial Risk

In this section we have discussed integration of Spark pipeline for preprocessing, for example Ray for model training, and Kubernetes for deployment. Performance benchmarks include latency, accuracy, and scalability. For example one commonly used stack in Gen AI is AWS services like Lambda, S3, and SageMaker. Key challenges include regulatory compliance (explainability, bias), model robustness in high-volatility scenarios, and the integration

of LLMs for hybrid risk assessment. Additionally, reinforcement learning (RL) cab be incorporated for adaptive risk strategies using the piplelines proposed in this work.

5.1 Full-Stack Big Data Frameworks for Generative AI

A robust and scalable full-stack big data framework is essential for effectively deploying Generative AI models in financial risk management. Based on the current literature we propose framework consists of three key layers:

- **Data Engineering Layer**: Apache Spark, Delta Lake, and AWS Glue facilitate distributed financial data processing. Spark enables large-scale data transformations, while Delta Lake ensures ACID transactions and versioning for consistent and reliable data. AWS Glue automates ETL workflows, allowing seamless integration across different data sources, including on-premise databases and cloud storage solutions like Amazon S3.
- **Model Training & Serving**: Ray with PyTorch/TensorFlow provides a scalable and distributed AI training environment. Ray Tune enables hyperparameter optimization, while Ray Serve allows efficient model deployment. Kubernetes orchestrates containerized model training workflows, ensuring elasticity and fault tolerance. AWS SageMaker streamlines the end-to-end ML lifecycle, offering managed Jupyter notebooks, AutoML capabilities, and model monitoring features. Additionally, multi-GPU training with AWS EC2 instances accelerates deep learning workflows.
- **Real-Time Risk Inference**: Kafka + Spark Streaming enable live risk assessment by processing high-frequency market data with low latency. Kafka acts as a high-throughput message broker, ensuring real-time event streaming, while Spark Streaming applies complex transformations for risk computations. ONNX Runtime optimizes AI inference by accelerating model execution across different hardware backends, including CPUs, GPUs, and TPUs. AWS Lambda provides a serverless execution environment, automatically scaling inference workloads based on demand and reducing infrastructure overhead.

This integrated framework ensures that financial institutions can leverage GenAI for risk modeling while maintaining efficiency, scalability, and regulatory compliance.

5.2 System Properties Comparison: Impala vs. Hive vs. Presto

Big data architectures supporting Generative AI in financial risk management require high-performance querying solutions. This section compares three distributed SQL engines—Impala, Hive, and Presto—based on their suitability for GenAI-driven workloads.

Impala: Known for its low-latency SQL analytics, Impala is well-suited for interactive financial risk queries. It excels in real-time data exploration, making it useful for GenAI-based models that require immediate access to large structured datasets.

Hive: A batch-oriented SQL engine optimized for handling vast amounts of financial data. While slower than Impala, Hive integrates well with Hadoop and is effective for historical data analysis and large-scale training datasets for GenAI models.

Presto: A federated SQL query engine that allows querying across multiple heterogeneous sources. Its flexibility makes it particularly suitable for GenAI applications requiring integration of structured market data, alternative datasets, and real-time financial streams.

From a Generative AI perspective, Impala provides the speed necessary for real-time model inference, Hive supports large-scale data preprocessing for training AI models, and Presto's federated capabilities enhance multi-source data aggregation, crucial for financial risk modeling.

5.3 Challenges and Performance Bottlenecks

Challenges with Hive and Trino ODBC Views for Frontend Integration are discussed in this section. Implementing Hive and Trino ODBC views in a full-stack financial data architecture presents several challenges, particularly when integrating with frontend applications for real-time data access and visualization. These issues arise due to performance bottlenecks, compatibility constraints, and data consistency concerns.

One of the primary challenges is query latency when using Hive for large datasets. Hive is optimized for batch processing and lacks the low-latency characteristics required for interactive frontend applications. This can result in slow query execution times, particularly for dashboards that require high-frequency updates.

Trino, formerly PrestoSQL, provides better performance for interactive queries due to its distributed architecture. However, its efficiency depends heavily on the underlying data storage and indexing strategies. Suboptimal table formats or missing indexes can degrade performance significantly.

5.4 ODBC Driver Limitations and Data Consistency Issues

While ODBC drivers enable seamless communication between the frontend and backend databases, compatibility issues can occur. For instance:

• **Data Type Mismatches:** Certain complex data types in Hive and Trino do not map well to frontend visualization tools, leading to errors or incomplete data retrieval.

• **Connection Stability:** Persistent ODBC connections may experience timeouts or instability when handling concurrent user requests, affecting frontend responsiveness.

• **Authentication Complexities:** Configuring secure authentication mechanisms (e.g., Kerberos or SSL) for ODBC connections can be challenging, particularly in cloud-based environments.

Data synchronization between Hive and Trino views can be problematic in environments where data updates occur frequently. Inconsistent data versions may be presented to frontend applications due to the eventual consistency model adopted by distributed storage solutions.

Additionally, schema evolution in Hive tables can cause compatibility issues with existing Trino views, requiring manual intervention to update schemas and maintain query compatibility.

5.5 Recommendations for Mitigation

To address these challenges, the following strategies are recommended:

- **Query Optimization:** Implement partitioning, bucketing, and indexing strategies for Hive tables to improve query performance. Use columnar storage formats such as Parquet for efficient data retrieval.
- **Connection Pooling:** Employ connection pooling mechanisms to manage ODBC connections efficiently and reduce latency.
- **Data Caching:** Use caching layers (e.g., Redis or Memcached) to store frequently accessed query results, reducing the need for repeated backend queries.
- **Schema Management:** Adopt schema versioning practices and automated synchronization tools to handle schema evolution seamlessly.
- **Security Enhancements:** Configure secure ODBC connections with modern authentication protocols and encryption to ensure data integrity and security.

By addressing these challenges, financial institutions can achieve more efficient frontend integration, enabling real-time insights and better decision-making capabilities for financial risk management.

5.6 Sample Code for Hive and Trino ODBC Integration

To demonstrate effective strategies for Hive and Trino integration via ODBC connections, here are sample code snippets:

Python Code:

Python Script for Trino Query Execution

```
import trino


# Establish connection to Trino
conn = trino.dbapi.connect(
host='your_trino_host',
port=8080,
user='your_username',
catalog='hive',
schema='default',
)
    # Execute a sample query
cursor = conn.cursor()
cursor.execute("SELECT * FROM financial_data LIMIT 10")
    # Fetch and print results
for row in cursor.fetchall():
print(row)
```

```
        cursor.close()
    conn.close()
Optimized Hive Query Example
    -- Enable vectorization and efficient file format
    SET hive.vectorized.execution.enabled=true;
    SET hive.execution.engine=tez;
        SELECT year, SUM(transaction_volume) as total_volume
    FROM financial_transactions
    WHERE year >= 2020
    GROUP BY year;
ODBC Connection Configuration Example (Linux)
    [ODBC Data Sources]
    TrinoDSN=Trino
        [TrinoDSN]
    Driver=/usr/lib/trino/odbc/libtrinoodbc.so
    Host=your_trino_host
    Port=8080
    Schema=default
    UID=your_username
    PWD=your_password
```

These code examples illustrate practical techniques for querying and managing financial data using Hive and Trino through ODBC connections. Proper configuration and optimized query practices are essential for seamless integration and improved performance.

6. Introduction to Kubernetes

This section systematically reviews the latest advancements in AI agents within the Azure Kubernetes ecosystem. This literature review discusses quantitative insights, architectural decisions, and emerging trends based on key studies and documentation. AI agents and Kubernetes orchestration are revolutionizing cloud-based AI/ML workloads. Microsoft Azure provides multiple services and frameworks for deploying and managing AI-driven applications [21], [22], [23].

6.1 AI Agents and Agentic Workflows

AI agents enable automated decision-making in cloud environments. Recent research highlights their role in DevOps and progressive delivery [24]. The modular architecture of Azure AI Agent Service enhances software development [25]. Additionally, Kubernetes plays a crucial role in AI agent deployment [26].

6.2 AI/ML Orchestration in Azure Kubernetes

Azure Machine Learning streamlines AI model deployment [23]. Google Cloud's AI/ML orchestration on Kubernetes further enhances these capabilities [27]. Open-source AI stacks, such as Determined AI, provide additional flexibility for Kubernetes-based AI operations [28].

6.3 Tables Kubernetes Integration for AI Agents

The integration of Kubernetes with AI agents is essential for scalability. Azure Kubernetes Service (AKS) supports AI workloads with GPU acceleration [29]. Cilium enhances Kubernetes networking, enabling efficient AI deployments [30]. Furthermore, the AI Toolchain Operator simplifies model deployment [31].

6.4 AI-Driven DevOps and Cloud Automation

AI-powered DevOps automates Kubernetes cluster management [32]. Kubernetes enhances AI agent deployment [33], while frameworks like AIOpsLab enable autonomous cloud resilience [34].

6.5 Architectural Design Decisions for Kubernetes

Designing AI-enabled Kubernetes environments requires careful architectural considerations:

Shetty et al. [34] propose a modular framework leveraging the agent-cloud-interface for real-time fault detection and resolution.

Microsoft Azure [21] emphasizes the use of prompt flows for building and deploying generative AI models in cloud environments.

The "AI Toolchain Operator" [31] simplifies the deployment and management of OSS AI models on Azure Kubernetes Service (AKS).

Several references highlight significant performance improvements and operational efficiencies achieved by integrating AI agents with Kubernetes:

- According to Anand [35], organizations deploying autonomous AI agents on Kubernetes observed a 30% reduction in manual operational interventions.
- The integration of GPUs and TPUs in Kubernetes environments, as outlined by Google Cloud [27], has enabled a 50% improvement in training speeds for large AI models.
- Shetty et al. [34] report promising results from the AIOpsLab prototype, which achieved real-time fault detection rates exceeding 85% through agent-based cloud monitoring.

6.6 Pseudocode for Agent Workflow

Below is a simplified pseudocode representation of a Kubernetes-based AI agent workflow:

Python Code:

```
function MonitorAndHeal(cluster):
while True:
status = checkClusterHealth(cluster)
if status == "FAULT":
log("Fault detected. Initiating recovery.")
triggerRecoveryProcedure(cluster)
else:
log("Cluster healthy.")

function checkClusterHealth(cluster):
// Query metrics and analyze patterns
return "FAULT" if anomalyDetected else "HEALTHY"

function triggerRecoveryProcedure(cluster):
// Roll back or redeploy failed services
log("Recovery complete.")
```

This pseudocode aligns with the architecture proposed by Shetty et al. [34].

7. Conclusion

This paper outlines a structured approach for integrating Generative AI Full Stack Data Engineering Framework in financial risk modeling. The proposed full-stack framework discuses the issues with scalability, real-time processing, and regulatory compliance. Future work includes autonomous risk

management systems powered by AI and the use of multi-agent reinforcement learning for risk adaptation. The integration of Generative AI with Trino has opened new frontiers in data analytics but has also introduced performance bottlenecks. This review highlights key applications, challenges, and optimization strategies. Furthermore, the bidirectional relationship between Trino and Generative AI presents opportunities for enhanced performance and scalability. Azure's AI ecosystem, integrated with Kubernetes, significantly enhances AI/ML workload management. Future research should focus on standardizing AI agent deployment frameworks for cloud resilience. The integration of AI agents with Kubernetes has emerged as a key enabler for autonomous cloud operations. By adopting modular architectures and leveraging advanced orchestration tools, organizations can achieve significant operational efficiencies and resilience. Future work includes autonomous risk management independent Agents powered by AI.

References

[1] S. Joshi, "The Synergy of Generative AI and Big Data for Financial Risk: Review of Recent Developments," *IJFMR - International Journal For Multidisciplinary Research*, vol. 7, no. 1, doi: g82gmx.

[2] S. Joshi, "Implementing Gen AI for Increasing Robustness of US Financial and Regulatory System," *International Journal of Innovative Research in Engineering and Management*, vol. 11, no. 6, pp. 175–179, Jan. 2025, doi: 10.55524/ijirem.2024.11.6.19.

[3] S. Joshi, "Review of Gen AI Models for Financial Risk Management," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, no. 1, pp. 709–723, Jan. 2025, doi: 10.32628/CSEIT2511114.

[4] S. Joshi, "Enhancing structured finance risk models (Leland-Toft and Box-Cox) using GenAI (VAEs GANs)," *International Journal of Science and Research Archive*, vol. 14, no. 1, pp. 1618–1630, 2025, doi: 10.30574/ijsra.2025.14.1.0306.

[5] S. Joshi, *Agentic Gen AI For Financial Risk Management*. Draft2Digital, 2025.

[6] S. Joshi, "Advancing Financial Risk Modeling: Vasicek Framework Enhanced by Agentic Generative AI by Satyadhar Joshi," *Advancing Financial Risk Modeling: Vasicek Framework Enhanced by Agentic Generative AI by Satyadhar Joshi*, vol. Volume 7, no. Issue 1, January 2025, Jan. 2025, doi: 10.56726/IRJMETS66819.

[7] S. Joshi, "A Literature Review of Gen AI Agents in Financial Applications: Models and Implementations," *International Journal of Science and Research (IJSR)*, doi: https://www.doi.org/10.21275/SR25125102816.

[8] S. Joshi, "Leveraging prompt engineering to enhance financial market integrity and risk management," *World Journal of Advanced Research and Reviews*, vol. 25, no. 1, pp. 1775–1785, 2025, doi: 10.30574/wjarr.2025.25.1.0279.

[9] S. Joshi, "Review of Data Engineering and Data Lakes for Implementing GenAI in Financial Risk," in *JETIR*, Jan. 2025.

[10] S. Joshi, "Agentic Generative AI and the Future U.S. Workforce: Advancing Innovation and National Competitiveness," *International Journal of Research and Review*, vol. 12, no. 2, 2025.

[11] S. Joshi, "Satyadharjoshi." Github Online Account https://github.com/satyadharjoshi

[12] R. Sethi *et al.*, "Presto: SQL on Everything," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, Macao, Macao: IEEE, Apr. 2019, pp. 1802–1813. doi: 10.1109/ICDE.2019.00196.

[13] "Generative AI and Starburst: Pioneering natural language interfaces for data exploration Starburst." https://www.starburst.io/blog/generative-ai-and-starburst-pioneering-natural-language-interfaces-for-data-exploration/.

[14] "Aerospike Connect for Trino Aerospike Documentation." https://aerospike.com/docs/connect/trino, Nov. 2024.

[15] "Query big data with resilience using Trino in Amazon EMR with Amazon EC2 Spot Instances for less cost AWS Big Data Blog." https://aws.amazon.com/blogs/big-data/query-big-data-with-resilience-using-trino-in-amazon-emr-with-amazon-ec2-spot-instances-for-less-cost/, Oct. 2023.

[16] "Presto: SQL on Everything - Meta Research," *Meta Research*. https://research.facebook.com/publications/presto-sql-on-everything/.

[17] "What Is Trino And Why Is It Great At Processing Big Data," *DEV Community*. https://dev.to/seattledataguy/what-is-trino-and-why-is-it-great-at-processing-big-data-8pc, Dec. 2021.

[18] D. Rangarao *et al.*, "Simplify Your AI Journey: Hybrid, Open Data Lakehouse with IBM watsonx.data."

[19] T. P. Morgan, "The Perfect AI Storage: Trino From Facebook And Iceberg From Netflix?" *The Next Platform*. https://www.nextplatform.com/2024/04/30/the-perfect-ai-storage-trino-from-facebook-and-iceberg-from-netflix/, Apr. 2024.

[20] J. Odmark, "What's the Difference Between Trino and PrestoDB?" *Pandio*. https://pandio.com/difference-between-trino-and-prestodb/, Mar. 2023.

[21] "Azure AI Foundry - Generative AI Development Hub Microsoft Azure." https://azure.microsoft.com/en-us/products/ai-foundry.

[22] "Azure AI Foundry - Generative AI Development Hub Microsoft Azure." https://azure.microsoft.com/en-au/products/ai-foundry.

[23] "Generative AI in Azure Machine Learning Microsoft Azure." https://azure.microsoft.com/en-us/products/machine-learning/generative-ai.

[24] "AI Agents and Agentic Workflow for DevOps and Progressive Delivery." https://www.xenonstack.com/blog/ai-agents-devops.

[25] J. MSV, "A Developer's Guide to Azure AI Agents," *The New Stack*. Feb. 2025.

[26] "Kubernetes For AI Agents Restackio." https://www.restack.io/p/agent-architecture-answer-kubernetes-ai-agents-cat-ai.

[27] "AI/ML orchestration on GKE documentation," *Google Cloud*. https://cloud.google.com/kubernetes-engine/docs/integrations/ai-infra.

[28] "Deploy on Kubernetes — Determined AI Documentation." https://docs.determined.ai/setup-cluster/k8s/_index.html.

[29] "Unlocking the Power of GPUs for AI and ML Workloads on Azure Kubernetes Services - The series," *Wesley Haakman*. https://www.wesleyhaakman.org/unlocking-the-power-of-gpus-for-ai-and-ml-workloads-on-azure-kubernetes-services-the-series/, Oct. 2024.

[30] "Cilium in Azure Kubernetes Service (AKS) - Isovalent." https://isovalent.com/blog/post/cilium-aks/, May 2023.

[31] schaffererin, "Deploy an AI model on Azure Kubernetes Service (AKS) with the AI toolchain operator (preview) - Azure Kubernetes Service." https://learn.microsoft.com/en-us/azure/aks/ai-toolchain-operator, Nov. 2024.

[32] "How generative AI could aid Kubernetes operations," *InfoWorld*. https://www.infoworld.com/article/3626661/how-generative-ai-could-aid-kubernetes-operations.html.

[33] "From Zero to Kubernetes in No Time: Thanks to AI Agents! by Vuyo Mhlotshane Feb, 2025 Medium." https://medium.com/@cmhlotshane/from-zero-to-kubernetes-in-no-time-thanks-to-ai-agents-4a500feee240.

[34] M. Shetty *et al.*, "Building AI Agents for Autonomous Clouds: Challenges and Design Principles." arXiv, Jul. 2024. doi: 10.48550/arXiv.2407.12165.

[35] V. Anand, "Autonomous Agentic AI for Kubernetes (open-source sw stack)," *Medium*. Dec. 2024.