# LEVERAGING METADATA, LINEAGE, AND AI AGENTS FOR INTELLIGENT REFACTORING IN CLOUD MIGRATIONS

Selvakumar Kalyanasundaram

Texas, USA

inboxofselva@gmail.com

## Abstract

As organizations increasingly adopt cloud computing to modernize their infrastructure, cloud migration projects are becoming a critical aspect of enterprise IT strategy. However, successful cloud migrations require more than simply transferring data and applications from on-premises environments to the cloud. They demand a deep understanding of data assets, dependencies, and processes. Metadata management and data lineage emerge as pivotal enablers for achieving efficient, secure, and risk-mitigated cloud migration. This paper explores the role of metadata and lineage in cloud migration initiatives, practical implementation strategies using AI methodology.

**Keywords:** Metadata, AI, LLM, Cloud Migration, Data Lineage

## Introduction

Most organizations are increasingly moving towards cloud-based IT infrastructure to enhance cost optimization, scalability, operational flexibility, and to avail advanced capabilities such as real-time analytics and AI-powered insights. While cloud environments offer transformative potential, achieving these benefits depends heavily on executing migrations with planning, visibility, and automation. In practice, many cloud migration efforts are hindered by critical challenges—data loss, inconsistent transformations, regulatory compliance gaps, and unplanned downtime. These issues frequently stem from inadequate visibility into legacy data ecosystems and insufficient planning around dependencies, data quality, and structural complexity.

Legacy systems such as Teradata typically include deeply nested schemas, undocumented transformations, embedded procedural logic, and dependencies across analytical and operational workloads. Migrating such systems—especially in regulated domains like healthcare—requires far more than a lift-and-shift approach. Among the common migration strategies (Rehost, Replatform, Refactor, and Replace), this paper focuses on **Refactoring**, wherein legacy components are re-engineered to fully leverage cloud-native features. This process is best enabled by a **metadata-driven migration strategy**, augmented by AI agents capable of automating lineage extraction, pipeline generation, SQL translation, and optimization.

## Literary Review:

As organizations increasingly shift from on-premises data warehouses to cloud-native platforms, the importance of structured, metadata-driven migration strategies has been extensively highlighted in both academic and industry literature. Studies by Gartner [1] and Deloitte [2] emphasize that metadata management is central to modern cloud migration efforts, enabling visibility into data assets, dependencies, and quality. Metadata helps mitigate migration risks such as data loss, transformation inconsistency, and regulatory noncompliance by providing context-aware insights into source systems, a concept foundational to the data modernization frameworks employed by large enterprises.

In particular, the role of data lineage has evolved from a governance tool to a functional enabler of safe and scalable migration. Research by Manta and Collibra [3][4] shows that column-level lineage not only enhances transparency but also supports automated impact analysis—critical for regulated sectors like healthcare. Further work by Khalilian et al. [5] explores how lineage graphs can be integrated with DataOps to dynamically adjust pipelines in response to schema changes or transformation failures, a capability increasingly demanded in continuous integration/continuous delivery (CI/CD) environments.

The application of Artificial Intelligence (AI)—especially Large Language Models (LLMs)—to cloud migration is an emerging frontier. Early efforts focused on rule-based metadata mapping tools, but more recent research demonstrates the ability of LLMs to infer semantic relationships, extract mappings from procedural SQL, and generate transformation logic automatically. Bansal et al. [6] proposed the use of transformer-based models for ETL code synthesis, showing promise in automating repetitive and complex migration tasks. Moreover, efforts like Microsoft's AutoML for Data Engineering [7] and Google Cloud's Dataform AI Assistant [8] provide practical examples of AI agents interpreting metadata and logs to optimize pipeline generation and cost modeling.

Healthcare-specific migration frameworks have also been examined. The work by Verhoef et al. [9] outlines a phased migration strategy that prioritizes sensitive datasets (e.g., PHI, PII), suggesting the use of scoring matrices incorporating metadata such as usage frequency, access patterns, and compliance flags. This aligns with the scoring approach discussed in this paper, where AI agents synthesize structured and unstructured metadata to automate prioritization and refactoring decisions.

Finally, security and governance literature has highlighted the need for encryption-aware metadata tagging and fine-grained access controls during migration. Solutions like Google BigQuery's column-level access policies [10] and AWS Lake Formation's data tagging architecture reinforce the criticality of integrating metadata classification and lineage with IAM enforcement layers to maintain regulatory compliance in cloud environments.

In summary, existing literature supports the core tenets of our implementation framework: metadata serves as the backbone for intelligent migration; AI and LLM agents can significantly enhance automation, transformation, and lineage tracing; and incorporating security-aware metadata enables regulatory alignment, especially in sensitive domains like healthcare. However, this paper advances the state of the art by demonstrating how these concepts coalesce into an operational strategy that bridges planning, automation, and post-migration optimization—thus transforming cloud migration from a tactical necessity into a strategic driver of innovation.

## Metadata Foundations for Refactoring

A structured metadata strategy serves as the foundation for intelligent migration and refactoring. Metadata is categorized as follows:

- **Structural Metadata**: Includes schemas, tables, columns, data types, and integrity constraints. These are essential for schema conversion and refactoring to BigQuery's native structures such as **partitioned** or **clustered tables** in a 3NF or flattened model.

- **Business Metadata**: Describes data semantics, field definitions, SLA agreements, domain ownership, and regulatory classifications (e.g., PHI, PII). This ensures contextual accuracy in data transformation and access control enforcement.
- **Usage Metadata**: Captures operational insights like query frequency, last accessed timestamps, pipeline load cycles, and user activity. This data informs prioritization and tiering decisions—enabling cost-efficient allocation of compute and storage.
- **Lineage and Dependency Metadata**: Tracks upstream and downstream data flows, ETL pipelines, materialized views, and transformation logic. Lineage ensures continuity and provides safeguards against breaking dependent services during schema or logic changes.
- **Quality Metadata and Statistics**: Includes null counts, data distributions, constraint violations, and historical anomalies. Quality metrics support pre-migration audits and post-migration validation, helping maintain trust and accuracy in migrated datasets.

**AI Agent Integration for Automation and Traceability**

Modernizing with agility requires more than static metadata—it demands intelligent agents capable of interpreting and acting upon metadata. **LLM-powered AI agents** play a critical role across the migration lifecycle. When metadata from legacy systems is fed into these agents using structured prompts, they can:

- Derive **semantic mappings** between legacy and target schemas
- Convert procedural SQL and stored procedures into **BigQuery-compatible SQL**
- Generate **human-readable summaries** of tables, domains, and relationships
- Construct **lineage graphs** at table and column level
- Recommend **refactoring approaches** (e.g., flattening nested structures, optimizing joins)
- Flag columns requiring **encryption** or fine-grained **access control policies**

These agents can also assist in building automated **data pipelines** using tools such as Apache Airflow, Dataform, or dbt, guided by mapping rules and lineage paths. Furthermore, lineage is extended from **source to ingestion pipeline to consumption layer**, enabling complete traceability.

**Security, Compliance, and Prioritization**

In healthcare environments, compliance with HIPAA and similar regulations is non-negotiable. As part of metadata tagging, AI agents flag columns containing PHI/PII and propagate encryption tags—both **at rest** and **in transit**—through the pipeline. Once in BigQuery, **fine-grained access control** is enforced via IAM, row-level and column-level security policies.

Table prioritization is achieved via an AI-generated **scoring matrix**, incorporating factors such as usage frequency, data sensitivity, schema complexity, last access date, and downstream dependency count. These scores guide phased migration strategies:

- **Phase 1**: High-value, sensitive, and business-critical tables (e.g., Claims, EHR)
- **Phase 2**: Operational and reference datasets
- **Phase 3**: Archival and decommissioned data

## Metadata-Driven Migration Methodology

### 1. Data Discovery & Data Profiling

Identify all data assets that need to be migrated. Data discovery should include all hidden or orphaned datasets. Some data might be stale. Some might be redundant. Some might be archive in different format. As they are legacy data , proper documents won't be available. So users struggle to find the data from the large set of datasets. Sever If the data been stored or archived in a specific metadata format. Several tools are available in market Alation, MANTA, Collibra data intelligence platform, Automate metadata extraction using tools like AWS Glue, Azure Purview, or SQL profiling. Extract schemas, constraints, usage stats, lineage, and field definitions.

### 2. Data Classification and Prioritization

Combine usage and business metadata to identify critical, sensitive, and cold data. The structure and naming standards might be hard for the teams to understand. Naming can be reformatted based on source and line of business / domain, ingestion type etc., Based on the requirement, the migration of the data can be prioritized.

AI agents are emerging as powerful tools for classifying the data. It enables the enterprise to make informed decision on data about whether to retain, refactor, or retire data assets. As enterprises face rapidly expanding volumes of data across the complex legacy environment, manual classification would be unmanageable. AI agents address this by leveraging a combination of machine learning (ML), natural language processing (NLP), and metadata analysis to autonomously analyze the datasets and determine their relevance, quality, usage frequency, and compliance risk. These agents use the metadata, access logs, ingestion scripts, data quality metrics and derive a comprehensive understanding of each dataset's lifecycle context. By applying dynamic scoring models trained on historical decisions and human feedback, AI agents can accurately predict classification categories. NLP techniques further enhance this process by interpreting business glossaries, table descriptions, and documentation to understand semantic meaning, reducing redundancy and identifying critical assets. Additionally, conversational interfaces powered by large language models (LLMs) allow AI agents to engage with data stewards or subject matter experts, gathering contextual feedback and improving classification accuracy over time. In large-scale projects such as cloud migrations or data warehouse modernization, AI agents have successfully classified thousands of tables, identifying low-value or dormant data for retirement, outdated structures for refactoring, and high-value assets for retention. This not only optimizes storage and processing costs but also strengthens data governance, ensures regulatory compliance, and accelerates decision-making. As these agents evolve through reinforcement learning and integration with governance platforms, they offer scalable, adaptive, and transparent data lifecycle management solutions for enterprises navigating increasingly data-intensive operations.

### 3. Mapping & Transformation

Accurate data lineage is essential for understanding the flow, transformation, mapping, and dependencies of data from source systems to target tables, particularly during large-scale modernization efforts such as cloud migration. Traditionally, lineage documentation captures mappings and transformation logic across systems. However, such documentation is often outdated, incomplete, or misaligned with the actual codebase running in production. This misalignment poses significant risks to data quality, compliance, and migration accuracy. To obtain up-to-date lineage, organizations typically rely on manual code reviews or ingestion log analysis—both time-consuming and error-prone tasks. An alternative and scalable approach is to leverage AI agents to automate the discovery and validation of data lineage. By processing ingestion logs over a defined historical window and analyzing ETL or ELT scripts, AI agents can extract transformation logic, frequency of execution, and source-target relationships. While log parsing and lineage derivation can be complex due to inconsistent formats or custom pipeline logic, AI agents augmented with prompt engineering and trained on relevant patterns can significantly simplify this process. These agents can be fine-tuned to recognize dataflow patterns, identify upstream and downstream dependencies, and validate lineage accuracy by comparing against metadata repositories or runtime environments. Moreover, with integration into source control systems and orchestration tools, AI agents can dynamically map lineage across multiple systems, flag discrepancies, and continuously update lineage graphs. This capability is particularly valuable when migrating tables and workflows from legacy systems to cloud platforms, as it ensures the completeness and consistency of mapping. Ultimately, AI-driven lineage extraction not only accelerates the migration process but also strengthens data governance, traceability, and audit readiness across the enterprise.

## 4. Automated Pipeline Deployment

Automating the generation of pipeline code through metadata-driven frameworks and ETL tools has become a critical enabler in modern data platform engineering, especially in the context of legacy system modernization and cloud migration. Rather than manually designing and developing each ingestion or transformation pipeline, organizations can leverage structured metadata—capturing source systems, transformation logic, and mapping rules—to dynamically generate reusable ETL/ELT code. This approach not only reduces development effort and cost but also enforces architectural consistency and accelerates deployment cycles. To further enhance this automation, AI agents can be integrated into the pipeline generation workflow, enabling intelligent parsing and interpretation of legacy SQL code, stored procedures, and batch scripts for automated conversion into target cloud-native constructs. For instance, an AI agent can ingest legacy SQL, interpret its logic, and generate corresponding SQL or pipeline definitions tailored for cloud data platforms such as Google BigQuery, AWS Redshift, or Azure Synapse, based on a user-specified target project, schema, or dataset.

A critical component in this process is the handling of schema evolution and mapping discrepancies between legacy and modern systems. AI agents, trained with prompt engineering and domain-specific models, can utilize a cross-reference mapping between legacy tables and cloud equivalents—either user-provided or automatically derived by correlating data lineage across environments. By aligning the lineage metadata from on-premise systems with that of the cloud environment, the AI agent can infer semantic matches between columns, validate transformation logic, and resolve naming conflicts, thereby constructing an accurate and executable target SQL or orchestration pipeline. Furthermore, AI agents can support continuous integration workflows by auto-generating DataOps pipelines, including data quality checks, schema validation, and deployment configurations. This AI-assisted conversion and orchestration pipeline not only accelerates the modernization journey but also ensures semantic integrity, reduces manual errors, and provides traceability for audits and compliance. Ultimately, integrating AI agents into metadata-driven code conversion frameworks transforms modernization into a more autonomous, scalable, and intelligent process, aligning well with enterprise goals of agility and cloud-first architecture. [3]

## 5. Validation & Reconciliation

Using metadata to drive row count comparison, checksum validation, and lineage tracing has become a foundational best practice in data migration and modernization initiatives. This metadata-driven validation approach ensures data accuracy, completeness, and traceability—especially during the transition from legacy systems such as Teradata to modern cloud platforms like Snowflake, Google BigQuery, or Azure Synapse. Structured pre- and post-migration checks leverage metadata tables that store source and target table mappings, primary keys, checksum columns, row count thresholds, and validation tolerances. During each iteration of data load in the cloud environment, these checks automatically compare source and target row counts and checksums, ensuring that the migrated data aligns with expected patterns and volumes.

To further scale and optimize this validation framework, AI agents—particularly those powered by large language models (LLMs)—can be deployed to interpret and reason over metadata and log files, dynamically generating validation queries and identifying anomalies. These LLM-based agents can extract business rules, primary key definitions, and transformation logic from legacy scripts and metadata repositories, then formulate validation logic for checksum and row count consistency across platforms. They can also assist in mapping data types, normalizing schema differences, and inferring lineage paths, improving the reliability of cross-system validation.

Moreover, these agents can intelligently manage validation scope by recommending selective sampling strategies or threshold-based validations based on data volume, pipeline maturity, or historical stability trends—thus optimizing compute resource usage during heavy cross-platform comparisons. As pipelines stabilize, LLM agents can be configured to flag when full validation can be deprecated or replaced by lighter, periodic audits, based on statistical drift analysis or anomaly detection patterns. By embedding AI-driven automation into the validation lifecycle, organizations can reduce operational overhead, improve data trust, and ensure seamless transitions from legacy data environments to modern, cloud-native platforms.

## 6. Post-Migration Monitoring & Optimization

Post-migration monitoring and optimization are essential for sustaining data quality, governance, and cost-efficiency in modern data platforms. Once data assets are transitioned to cloud environments such as Snowflake, Google BigQuery, or Azure Synapse, ongoing analysis of usage metadata becomes critical for automating actions like tiering, archiving, or decommissioning underutilized datasets. Metadata such as query logs, access patterns, record counts, ingestion frequency, partition usage, and clustering effectiveness can be collected using native tools—for instance, **Teradata DBQL**, **Snowflake's ACCESS_HISTORY**, or **BigQuery's INFORMATION_SCHEMA** views. This metadata enables granular tracking of how data is consumed, transformed, and maintained over time.

To operationalize this at scale, organizations can implement a metadata-driven framework that assigns classification tags to tables and columns based on application relevance, data types, ownership, and regulatory requirements. AI agents, particularly those powered by **LLMs fine-tuned for data engineering tasks**, can automate the interpretation and classification of this metadata. These agents can continuously monitor access logs, detect usage anomalies, and suggest data lifecycle actions such as transitioning cold data to archival storage tiers, initiating table decommissioning, or refining partitioning strategies to optimize query performance. AI agents can also maintain **data quality scores**, detect schema drift, and validate **forward lineage**—ensuring that downstream processes remain consistent when changes occur upstream, such as the addition, removal, renaming, or data type modification of attributes.

Furthermore, these AI agents can be embedded into orchestration frameworks using tools such as **Apache Airflow** for scheduling, **SQL-based metadata repositories** for rule application, and **Python-based rule engines** for executing optimization workflows. When integrated with enterprise data governance platforms like **Google Dataplex**, **Collibra**, or **MANTA**, AI agents enhance traceability and auditability, automatically logging metadata changes and lineage shifts. These logs can be collected and processed at daily or hourly intervals—depending on the ingestion cadence—to detect data loss, latency issues, or operational bottlenecks.

Importantly, AI agents can also automate the generation and validation of post-migration scripts that reconfigure partitioning, clustering, or indexing strategies based on observed usage trends. They can synthesize SQL and optimization logic based on metadata patterns, enabling continuous code refinement without human intervention. When impactful schema changes are detected, AI agents can generate automated impact analysis reports and notify affected stakeholders, ensuring proactive communication and minimizing operational risk.

By embedding AI agents into the metadata lifecycle, organizations can transform their post-migration monitoring into an intelligent, self-optimizing system—improving cost control, query performance, and regulatory compliance, while reducing manual overhead and ensuring ongoing data trust.

**Role of Data Lineage in Cloud Migration**

Data lineage serves as a foundational pillar in modern data architecture, capturing the end-to-end lifecycle of data—from its origin in source systems, through its various transformations, to its eventual destination in reporting, analytics, or operational platforms. In migration and modernization projects, particularly those involving critical infrastructure transitions from legacy systems to cloud platforms, accurate and complete data lineage is essential for ensuring data trust, continuity, and regulatory compliance.

Lineage enables teams to understand the upstream and downstream dependencies of datasets, facilitating comprehensive **impact analysis** when changes occur. This becomes especially critical during schema restructuring, transformation logic adjustments, or data consolidation in cloud environments. By visually and semantically comparing pre- and post-migration data flows, organizations can confirm that business processes remain functionally intact and that no critical relationships are broken in the transition. Lineage also plays a vital role in **verifying transformation logic**, ensuring that the semantic meaning of fields and business rules applied during ETL or ELT processes are faithfully preserved throughout the migration journey.

Moreover, in regulated industries such as healthcare and finance, lineage is indispensable for tracking the flow of **Personally Identifiable Information (PII)** and **Protected Health Information (PHI)** across data pipelines. It supports audit trails, consent management, and privacy compliance requirements such as HIPAA and GDPR by identifying how sensitive data propagates through systems and who has accessed or modified it at each stage. This traceability strengthens organizational governance and risk mitigation capabilities.

The integration of **AI agents**, especially those powered by **large language models (LLMs)**, is revolutionizing how lineage is extracted, validated, and operationalized. These agents can analyze ingestion logs, transformation scripts, stored procedures, and metadata repositories to automatically construct lineage graphs without the need for manual rule definition. By applying prompt engineering techniques and pre-trained semantic models, LLM-based agents can interpret business logic embedded in legacy SQL or procedural code, infer column-level lineage, and reconcile mappings across disparate platforms.

Furthermore, AI agents can detect lineage inconsistencies between environments, flag undocumented transformations, and enrich lineage graphs with contextual metadata such as data classifications, security tags, and ownership. They also support dynamic validation, continuously monitoring for changes in source schema or transformation logic and assessing the downstream impact—automatically generating alerts, remediation steps, or documentation updates.

By leveraging AI-driven lineage tracing, migration teams can move beyond a simplistic “lift-and-shift” model toward a **controlled, governed, and traceable modernization approach**. This not only minimizes operational risk but also enables organizations to **fully leverage cloud-native capabilities**, such as metadata-aware optimization, data cataloging, and privacy-aware compute governance. As lineage becomes more automated, real-time, and intelligent, it will serve not only as a compliance tool but as a strategic asset in enterprise-wide data lifecycle management.

**Implementation Strategy with a sample use case:**

In a large-scale enterprise modernization initiative within the healthcare sector, the migration of a Teradata-based on-premise data warehouse to Google BigQuery was undertaken using a metadata-driven and AI-assisted strategy. Given the complexity of the legacy ecosystem—comprising thousands of tables spanning clinical, claims, financial, and EHR domains—a systematic, intelligent, and domain-aware migration process was essential. The implementation was designed to ensure data integrity, privacy compliance, prioritization, and optimal use of cloud-native features post-migration.

**1. Metadata-Driven Data Discovery and Domain Classification**

The migration process began with **data discovery** using Teradata system metadata. This included extracting technical metadata such as table names, column data types, data sizes, row counts, and operational metadata such as last access timestamps, frequency of use, and access patterns. Business metadata was tagged by identifying key terms within table and column names—such as “Patient,” “Claims,” “EHR,” or “Revenue”—to assign domain relevance. Sensitive fields containing **PHI (Protected Health Information)** or **PII (Personally Identifiable Information)** were automatically flagged using pattern-based tagging and NLP-enhanced column name analysis.

To enhance domain identification and classification accuracy, **Large Language Model (LLM)-powered AI agents** were introduced. The complete metadata corpus was fed into LLMs via structured prompts to extract semantic insights and cluster tables by business domains. These agents provided **human-readable summaries** for each table and column, enabling stakeholders to validate classification at scale. Additionally, ingestion source tagging was performed to group tables by their upstream systems, further facilitating orchestration and dependency mapping.

**2. AI-Augmented Prioritization with Scoring Matrix**

To prioritize tables for migration, a **multi-dimensional scoring matrix** was developed based on metadata features such as table usage, size, sensitivity, data quality, and lineage complexity. The scoring matrix was computed using both rule-based logic and AI-driven techniques. Initial scoring metrics included:

| Tablename | ColumnSensitivity | ColumnCount | Complexity | Tablesize | AccessFrequency |
|---|---|---|---|---|---|
| Patient_table | Yes | 115 | 20 | 11G | 7 |

Derived scores:

```
df['PriorityScore'] = (
    0.3 * df['ColumnCount'] +
    0.2 * Boolean(df['ColumnSensitivity']) +
    0.2 * df['AccessFrequency'] +
    0.15 * df['Tablesize']
)
```

In parallel, historical prioritization decisions (where available) were used to train an **XGBoost classifier**, enabling **AI-based prediction** of migration priorities. This model integrated metadata features to forecast whether a table should be categorized as **High**, **Medium**, or **Low Priority**.

### 3. Migration Grouping and Roadmap Creation

Based on priority and domain, tables were grouped using the **Traditional 'R' migration strategy**, focusing only on Refactor + Replatform along with Archive. A **migration roadmap** was constructed in phases:

- **Phase 1**: High-priority, business-critical domains (e.g., Claims, EHR)
- **Phase 2**: Operational/reference data and feeder systems
- **Phase 3**: Archived, backup, and deprecated datasets

### 4. AI-Assisted Lineage and Mapping Extraction

To ensure consistency and traceability, **end-to-end data lineage** was derived at the table and column level. Legacy SQL and stored procedure logs were extracted from Teradata using a defined historical window. These scripts were fed into LLM-based AI agents with prompts engineered to:

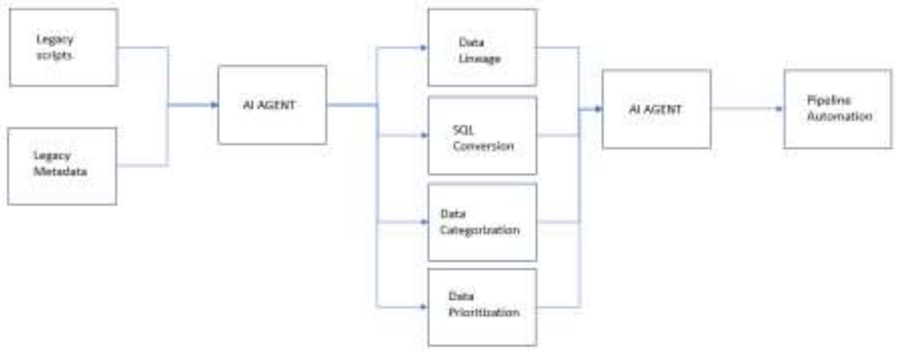
- Extract source and target tables
- Identify joins, filters, calculations
- Parse transformation logic
- Generate lineage chains and mapping rules

This AI-extracted lineage was correlated with pre-existing metadata repositories to build comprehensive **column-level lineage graphs** and transformation maps. These were further used to validate post-migration data pipelines and support **impact analysis**, particularly for sensitive data fields.

### 5. Operationalization and Monitoring

The final stage involved automating the migration process using **Apache Airflow** for pipeline scheduling, **SQL-based metadata repositories** for tracking, and **Python rule engines** for data quality enforcement. Metadata integration with tools like **Google Dataplex**, **MANTA**, and **Collibra** ensured governance and audit readiness.

AI agents continued to monitor **post-migration performance**, analyzing usage metadata (e.g., query frequency, pipeline failures, execution times) and suggesting optimizations such as **partitioning strategy refinement**, **cost reduction via tiering**, and **automated decommissioning** of stale datasets.



Sample code to extract the lineage

```
Context =”””Context is Following are the details of the {table_name} and the ingestion script
is {inp_sql}“
Task=” provide the lineage of the {table_name} to the source table
        If there is a transformation to derive the target table column provide the transformation
logic in transformation attribute of output “
Example=”’json
  {table_name:”
Columns: [
{column_name:”,
Source_table:<provide the source table or file>”
Source_column:<provide the column name from source>”,
Transformation:<>
Business_logic:<>
Joins_used:<>””

Response = query_claude(Context,max_tokens=60000,bot_engine=”claude-4-
sonnet”,chat_history=None,thinking_budget=15000,response_format:Union[str,Any]=”text”)
```

In large-scale data warehouse modernization efforts, the generation of ingestion and transformation pipelines can be significantly accelerated by feeding structured mapping logic—derived from metadata, data profiling, and lineage artifacts—into AI agents. These LLM-powered agents interpret source-to-target mappings, transformation rules, data types, and business rules to generate automated ETL/ELT pipeline code tailored for cloud-native execution environments. For example, JSON or CSV-based mapping documents describing how Teradata source fields relate to BigQuery target fields—including join logic, filters, and derived columns—can be ingested by an AI agent and translated into executable BigQuery SQL, dbt models, or Apache Beam DAGs.

In parallel, legacy SQL scripts and stored procedures written for platforms like Teradata or SQL Server can be automatically converted to BigQuery-compatible SQL using prompt-engineered LLMs. These agents can parse procedural logic, temporary table usage, cursor-based iteration, and variable declarations, and intelligently refactor them using idiomatic constructs such as WITH clauses, analytic functions, and native BigQuery UDFs. In many cases, the AI agent can also optimize the translated code by applying best practices for partitioning, clustering, and table decorators.

A critical component of this modernization is the ability to track end-to-end data lineage—from source systems to ingestion pipelines, transformation layers, and consumer-facing tables or views. LLM-based agents can be prompted with ingestion scripts, transformation logic, and orchestration metadata (e.g., from Airflow or Dataform) to construct column-level lineage graphs. These graphs document how individual columns evolve, merge, or split across pipeline stages, providing valuable transparency for both business stakeholders and technical teams.

Security and compliance are tightly integrated into this process. Columns tagged as sensitive, encrypted, or regulated (e.g., PHI/PII) are flagged during lineage derivation and marked with corresponding encryption-at-rest and in-transit annotations in the metadata catalog. The migrated datasets in BigQuery are provisioned with fine-grained access controls, leveraging IAM policies and row/column-level security rules to enforce least privilege access. AI agents can further assist in auditing these policies, detecting policy drift or access violations based on actual query patterns.

Furthermore, usage metadata—such as table access frequency, query volume, and downstream dependency count—is used in conjunction with lineage data to prioritize which assets require stricter governance, which can be archived, and which benefit from additional optimization (e.g., partition pruning, sharding). LLMs can also be prompted to generate natural language summaries of lineage paths, making lineage more consumable to non-technical stakeholders and enhancing collaborative data governance.

## Conclusion

In summary, the integration of AI agents powered by LLMs into the pipeline derivation, SQL translation, and lineage documentation process results in a more automated, secure, and auditable cloud migration. This reduces time-to-value, enhances regulatory compliance, and positions enterprises to leverage cloud-native features effectively while maintaining full transparency and traceability of data transformations.

## References

1. **Gartner. *Best Practices for Cloud Migration: Metadata Management and Data Governance*, 2022.**

2. **M. Black & D. Veroff, *Unlocking Enterprise Innovation in the Cloud: Strategy and Blueprinting for Healthcare Organizations*, Deloitte Consulting LLP, 2022.**

3. **MANTA. *Automated Lineage for Modern Cloud Architectures*, Whitepaper, 2021.**

4. **Collibra. *Data Intelligence Cloud for Cloud Migration Governance*, Collibra, 2023.**

5. **M. Khalilian, et al., “An Ontology-Based Data Lineage Framework for Adaptive Data Pipelines,” *Journal of Big Data*, vol. 7, no. 1, 2020.**

6. **S. Bansal, P. Rajpurkar, A. Ng, "Prompting AI Agents for ETL Automation in Data Warehousing," *IEEE Transactions on Big Data*, vol. 10, no. 2, 2023.**

7. **Microsoft Research. *AutoML for Data Engineering Tasks*, 2022.**

8. **Google Cloud. *Dataform AI Assistant: Transforming ELT Automation*, 2024.**

9. **R. Verhoef, T. Simons, et al., “A Risk-Aware Migration Strategy for Sensitive Healthcare Data,” *Health Informatics Journal*, vol. 26, no. 3, 2020.**

10. **Google Cloud. *Fine-Grained Access Control in BigQuery*, 2023.**