Building a High-Impact Anti-Money Laundering Data Engineering Portfolio Project: A Guide for Aspiring AML Data Engineers

I. Foundations: AML and Modern Data Engineering

Anti-Money Laundering (AML) is a critical function within financial institutions, aimed at preventing criminals from disguising illegally obtained funds as legitimate income. Data engineering plays an increasingly vital role in these efforts, providing the technological backbone for detecting and reporting suspicious activities. This section lays the groundwork by explaining the fundamentals of money laundering, the responsibilities of data engineers in this domain, and the objectives of the proposed portfolio project.

A. A Primer on Anti-Money Laundering (AML)

Money laundering is the process of making illegally-obtained proceeds appear legitimate. It typically involves three distinct stages, each presenting unique challenges and data footprints for detection systems.[1]

1. **Placement:** This initial stage involves introducing illicit funds into the financial system. Criminals often achieve this by breaking down large sums of cash into smaller, less conspicuous amounts that are then deposited into bank accounts or used to purchase monetary instruments like checks or money orders.[1] The primary goal is to move the funds from the direct association with the crime.
2. **Layering:** Once the funds have entered the financial system, the layering stage commences. This is often the most complex phase, involving a series of transactions designed to obscure the audit trail and dissociate the money from its criminal origin.[1] Common layering techniques include:
    - Electronic transfers between multiple accounts, often across different institutions and jurisdictions.[1]
    - Purchasing and reselling financial instruments such as stocks, bonds, or insurance products.[1]
    - Investing in high-value assets like real estate or luxury goods, which can then be sold.[1]
    - Utilizing international transactions, including foreign currency exchanges and offshore accounts, to make tracing more difficult.[1]
    - Masking illicit funds within legitimate business transactions, such as over/under-invoicing for goods and services or setting up fictitious loans.[1]
    - Exploiting newer financial technologies, including cryptocurrency trading and

emerging payment methods, due to perceived anonymity or weaker regulatory oversight.[1]

3. **Integration:** The final stage is the integration of the laundered funds back into the legitimate economy. At this point, the money appears to have originated from a lawful source, allowing criminals to use it without attracting suspicion.[1] Integration methods can include:
    - Transferring funds into legitimate bank accounts as purported business profits or dividends.[1]
    - Selling assets that were acquired during the layering stage.[1]
    - Creating fake employment and issuing salaries to the criminals or their associates.[1]
    - Investing in legitimate businesses or luxury items.

Understanding these three stages is fundamental for data engineers working in AML. Each stage generates different types of data and transaction patterns. An effective AML data pipeline must be designed to capture and analyze these diverse data footprints to identify suspicious activities indicative of any of these stages. The continuous evolution of laundering techniques, particularly in the layering phase with the adoption of cryptocurrencies and new payment methods [1], means that AML data engineering solutions must be inherently flexible. They need to be capable of incorporating new data sources and adapting detection logic as financial criminals devise new schemes. The increasing sophistication of these layering methods, such as complex international fund movements and the use of intricate corporate structures [1], also pushes the requirements for data analysis beyond simple rule-based checks, necessitating the use of advanced techniques like machine learning and network analysis to uncover subtle and hidden relationships.[5]

B. The Data Engineer's Role in Combating Financial Crime

Data engineers are the architects and builders of the systems that power modern AML efforts. Their primary responsibility is to design, implement, and maintain the infrastructure required to collect, store, process, and analyze the vast quantities of financial data generated by an institution.[7] This involves:

- Creating robust and scalable data pipelines to ingest data from various internal systems (e.g., core banking, transaction processing) and external sources (e.g., watchlists, adverse media feeds).
- Ensuring data quality, consistency, and reliability, which are paramount for accurate AML analysis and regulatory reporting.
- Transforming raw data into formats suitable for analysis, including the creation of meaningful features that can indicate risk.

- Implementing and managing data storage solutions, often involving data lakes and data warehouses.
- Enabling advanced analytics, including supporting machine learning model development and deployment for anomaly detection and risk scoring.[9]

A significant challenge often faced in large financial institutions is the existence of "siloed datasets" [10], where critical information resides in disparate, unconnected systems. This fragmentation can severely hamper effective AML analysis. Consequently, a key task for data engineers is to develop solutions that integrate and unify these varied data sources, providing a comprehensive and holistic view of customer activities and associated risks. Furthermore, the demand for "real-time visibility into risks" [10] is pushing AML data engineering towards incorporating stream processing technologies and near real-time analytical capabilities. While batch processing remains essential for historical analysis and model training, the ability to detect and react to suspicious activities as they happen is becoming increasingly critical for timely intervention.

C. Project Objective: Building an AML Suspicious Activity Detection System

The primary objective of this portfolio project is to design and implement a data engineering solution that simulates a real-world Anti-Money Laundering system. This system will focus on identifying suspicious financial transactions by leveraging a combination of the Big Data Hadoop ecosystem, AWS Cloud services, and the Snowflake Data Cloud platform. This project is specifically tailored to allow for the practical application and demonstration of skills in these technologies, as recommended.

The success of this project will hinge not only on the technical construction of the data pipeline but also on demonstrating an understanding of the underlying AML principles that drive the detection logic. For instance, when implementing specific detection rules within Snowflake, it will be important to articulate *why* a particular rule—such as one flagging the rapid movement of funds [11]—is relevant to identifying potential layering activities. The selection of technologies for this project reflects a modern, hybrid architectural approach prevalent in the financial industry. The Hadoop ecosystem, particularly Apache Spark, offers powerful capabilities for scalable batch processing and complex data transformations.[12] AWS provides the robust and flexible cloud infrastructure and a suite of managed services essential for building and deploying such a system.[13] Snowflake, as a cloud-native data warehouse, delivers high-performance analytics, flexible data modeling, and a powerful SQL engine, complemented by capabilities like Snowpark for in-database machine learning.[14] This

project will showcase the synergy between these technologies, where Hadoop/Spark might be used for initial large-scale ETL and data preparation, with the refined data then loaded into Snowflake for advanced analytics and AML rule application.

II. Project Blueprint: An AML Transaction Monitoring System

This section outlines the conceptual architecture and core idea for the AML transaction monitoring system. It will focus on developing a scalable pipeline to identify anomalous financial transactions, targeting specific, common money laundering typologies.

A. Conceptual Architecture: From Raw Data to Actionable AML Insights

Financial data processing, especially for AML, often employs a multi-layered architecture to manage the flow and transformation of data from its raw state to actionable insights.[13] This project will adopt a similar layered approach:

1. **Data Sources Layer:** This layer involves the initial acquisition of raw data. For this project, the primary source will be a synthetic financial transaction dataset (e.g., SAML-D).
2. **Data Integration & Storage (Data Lake):** Raw data will be ingested and stored in its original format in an AWS S3 data lake. S3 will serve as the landing zone.
3. **Processing Layer:** AWS EMR, utilizing Apache Hadoop and Apache Spark, will be responsible for processing the raw data from S3. This includes data cleansing, transformation, and feature engineering. The processed data will then be written back to S3 in an optimized format (e.g., Parquet) in a separate "processed" zone.
4. **Analytical Data Warehouse Layer:** Snowflake will serve as the central analytical data warehouse. Processed data from S3 will be loaded into Snowflake tables. This layer is where AML detection rules will be applied using SQL, and optionally, where machine learning models can be deployed using Snowpark.
5. **Presentation & Alerting Layer:** AWS QuickSight will connect to Snowflake to visualize flagged transactions, key risk indicators (KRIs), and provide a dashboard for hypothetical AML analysts. QuickSight can also be configured for basic threshold alerts.

Project Architecture Diagram:

Code snippet

```
graph TD
    A -->|Upload| B(AWS S3 - Landing Zone <br> s3://your-bucket/landing/);
    B -->|Read Raw Data| C{AWS EMR Cluster <br> (Hadoop, Spark, Hive)};
    C -->|PySpark: Cleanse, Transform, Feature Engineering, EDA| D(AWS S3 - Processed/Curated Zone <br> s3://your-bucket/processed/ <br> Format: Parquet);
    D -->|ETL: AWS Glue or Spark Connector| E(Snowflake Data Cloud <br> Tables: TRANSACTIONS, ALERTS);
    E -->|SQL Rules, Optional Snowpark ML| F(Snowflake: AML Alerts Table);
    F -->|Read Alert & Transaction Data| G(AWS QuickSight <br> AML Dashboard & Visualizations);
    C -->|Store Logs & Scripts| H(AWS S3 - Logs/Artifacts <br> s3://your-bucket/logs/ <br> s3://your-bucket/emr-artifacts/);
```

The strategic use of different technologies at various stages is a key aspect of modern data architectures. While cloud-native platforms like Snowflake offer powerful analytical capabilities [14], they often work in conjunction with systems like Hadoop and Spark. These are frequently employed for initial, cost-effective, large-scale data preparation and ETL before loading refined datasets into platforms like Snowflake for high-performance querying and analytics.[13] This project is designed to demonstrate proficiency in orchestrating such a synergistic, multi-system pipeline. Furthermore, even though this project utilizes synthetic data, adopting data governance best practices, such as maintaining clearly demarcated data zones (raw, processed, curated) within the S3 data lake [16], adds a layer of realism and demonstrates an understanding of constraints pertinent to regulated environments like AML, where data lineage and auditability are critical.[13]

B. Core Project Idea: Developing a Scalable Pipeline to Identify Anomalous Financial Transactions

The central aim of this project is to construct an end-to-end data pipeline capable of ingesting financial transaction data, processing it at scale, applying analytical techniques to identify potentially suspicious activities, and flagging these activities for review. This aligns directly with the core functions of an AML data engineering team and provides a platform to demonstrate skills across the entire specified technology stack.

A key characteristic of this project is its inherent scalability. The choice of AWS EMR for Hadoop/Spark processing and Snowflake for data warehousing ensures that the designed pipeline can, in principle, handle datasets far larger than the initial synthetic dataset used for development.[12] This is crucial because real-world financial

institutions deal with millions, if not billions, of transactions daily. The project can also be developed in phases, starting with the implementation of rule-based detection mechanisms, which are generally simpler to build and interpret. Subsequently, as an advanced extension, machine learning-based anomaly detection models can be incorporated [5], showcasing an ability to implement more sophisticated detection techniques.

C. Targeting Key AML Typologies

To make the project concrete and allow for focused development of detection logic, it will target several common money laundering typologies. These typologies represent patterns of behavior that are often indicative of attempts to misuse the financial system. Guidance from organizations like the Financial Crimes Enforcement Network (FinCEN) [18] and the Financial Action Task Force (FATF) [2], as well as industry groups like the Wolfsberg Group [23], provide extensive information on these patterns. For this project, the focus will be on typologies such as:

- **Structuring:** Executing multiple transactions just below mandatory reporting thresholds to avoid scrutiny.[1] For example, multiple cash deposits of $9,000 when the reporting threshold is $10,000.
- **Rapid Movement of Funds (Layering):** Characterized by funds being deposited into an account and then quickly withdrawn or transferred to other accounts, often in different jurisdictions, to obscure the source.[1]
- **Transactions Involving High-Risk Jurisdictions:** Conducting financial activities with counterparties or financial institutions located in countries known for weak AML/CFT regimes, high levels of corruption, or those under international sanctions.[10]
- **Unusual Transaction Patterns:** Deviations from a customer's established financial behavior, such as a sudden increase in transaction frequency or volume, or transactions that are inconsistent with the customer's known profile or business activities.[6]
- **Use of Multiple Accounts / Funnel Accounts:** Consolidating funds from multiple sources into a single account or, conversely, dispersing funds from one account to many, often as part of a layering scheme.[25]

It is important to recognize that AML typologies are not static; they continuously evolve as criminals adapt their methods to circumvent existing controls.[6] Therefore, while this project will focus on a defined set of typologies, a real-world AML system must be designed for ongoing updates and the incorporation of new detection logic based on emerging threats and intelligence. The ability to effectively detect these typologies is heavily dependent on the quality and completeness of the underlying

data [9] and, critically, on the features engineered from this data.[17] For instance, to identify transactions linked to high-risk jurisdictions, the dataset must contain accurate geographical information for both the sender and receiver of funds.

III. Data: The Fuel for Your AML Project

The availability and nature of data are foundational to any data engineering project, particularly in the AML domain. This section discusses the rationale for using synthetic data and recommends specific datasets suitable for this project.

A. The Case for Synthetic Data in AML Projects

Access to real financial transaction data for development and portfolio projects is severely restricted due to stringent privacy regulations (like GDPR, CCPA), legal frameworks, and overriding security concerns.[27] Financial institutions cannot readily share customer transaction data with external parties or even internal teams without extensive safeguards and anonymization, which itself can be complex and may strip out valuable analytical signals.

Synthetic data offers a practical and ethical solution to this challenge. These are artificially generated datasets designed to mimic the statistical properties, patterns, and complexities of real-world data without containing any actual Personally Identifiable Information (PII).[28] For AML projects, synthetic datasets can be specifically engineered to include various money laundering typologies, allowing developers to build and test detection mechanisms in a safe and compliant environment.[29] The quality of synthetic data is assessed based on its fidelity (how closely it mirrors real data behavior), privacy (assurance that no real identities are exposed), and utility (its effectiveness in supporting valid testing and model calibration).[30] Different generation techniques, such as agent-based simulations [31] or Generative Adversarial Networks (GANs) [30], are used to create these datasets, each with its own strengths in replicating complex real-world scenarios. While synthetic data effectively bypasses PII concerns [28], it's important to acknowledge a potential limitation: it may not always capture the most novel or rapidly emerging laundering techniques that have not yet been incorporated into the generation models. This means that models and rules developed solely on synthetic data may need further tuning and validation when deployed against actual production data.

B. Recommended Datasets

Several synthetic datasets are available that can be used for this AML project. The primary recommendation is the SAML-D dataset due to its specific design for AML

research and its public availability.

1. **Primary Recommendation: SAML-D Dataset**
    - **Overview:** The Synthetic Anti-Money Laundering Dataset (SAML-D) was created using a novel transaction generator designed to produce realistic financial transactions incorporating a diverse range of AML typologies.[27] It comprises 9,504,852 transactions, of which a small fraction (0.1039%) are labeled as suspicious, reflecting the imbalanced nature of real-world AML data.[27]
    - **Features & Typologies:** The dataset includes 12 core features and covers 28 distinct typologies (11 normal transaction patterns and 17 suspicious ones). These typologies were developed based on existing datasets, academic literature, and interviews with AML specialists.[27] Key features include transaction timestamps, sender and receiver account details, transaction amounts, payment types, sender and receiver bank locations, payment and received currencies, a binary indicator for suspicious transactions (Is_laundering), and a more granular Laundering_type field.[27]
    - **Access:** SAML-D is publicly available on Kaggle, making it easily accessible for this project.[27] The original research paper [31] provides valuable context on its generation and characteristics.
    - **Suitability:** Its explicit design for AML, inclusion of diverse and labeled typologies, representation of geographic risk factors (high-risk countries and payment types), and the presence of both binary and multi-class labels for suspicious activity make it highly suitable for developing and testing both rule-based and machine learning-based detection systems.[31] The Laundering_type feature is particularly valuable as it allows for more nuanced analysis and potentially the development of models that can suggest the specific *type* of suspicious activity detected.
2. **Alternative Recommendation: IBM Synthetic Data Sets for Core Banking and Money Laundering**
    - **Overview:** This dataset is part of IBM's broader family of synthetic datasets and is specifically curated for AML solutions.[28] It contains simulated banking transactions labeled for various financial crimes, including money laundering, check fraud, and Automated Push Payment (APP) fraud. The dataset captures detailed fraud scenarios and laundering activities, including account details and fund transfers.[28]
    - **Format and Access:** The datasets are typically provided as downloadable CSV and DDL files, ensuring compatibility with a wide range of databases and analytical tools.[28] Importantly, they are generated without using any real client

PII, alleviating privacy concerns.[28] Access may require inquiry through IBM channels, and the associated IBM Redpaper (referenced in [28]) would offer more comprehensive details on its structure and acquisition.
    - **Suitability:** Being specifically designed for AML by a major technology provider, this dataset is a strong alternative. It includes labeled data and aims to cover transaction types like global and cash transactions, which are often underrepresented in other datasets.[28]
3. **Other Potential Datasets (for Contextual Awareness):**
    - **AMLSim/AMLWorld:** AMLSim is generated using a publicly available repository, and the AMLWorld dataset is available on Kaggle.[31] These are often used in academic research but might require more extensive preprocessing compared to SAML-D or the IBM dataset.
    - **PaySim:** This dataset simulates mobile money transactions and includes fraud labels.[31] While primarily used for general fraud detection, its transaction-level data (including features like step, type, amount, sender/receiver identifiers, balances, and fraud flags [38]) could be adapted for certain AML scenarios. However, SAML-D or the IBM dataset are more directly aligned with AML typologies.

The choice between SAML-D and the IBM dataset may depend on the specific AML typologies one wishes to focus on and the ease of obtaining the data. SAML-D's immediate availability on Kaggle is a significant practical advantage for starting the project promptly. A critical characteristic of the SAML-D dataset is its realistic class imbalance, with only 0.1039% of transactions labeled as suspicious.[27] This mirrors real-world AML scenarios where fraudulent or suspicious transactions are rare events. Addressing this imbalance (e.g., through appropriate machine learning sampling techniques, using anomaly detection algorithms robust to imbalance, or focusing evaluation metrics on precision and recall rather than overall accuracy) will be an important aspect of the project, reflecting a common challenge in the field.

C. Understanding Dataset Features Relevant to AML

The features within the chosen dataset are the raw ingredients for building detection rules and training machine learning models. Using SAML-D as the primary example, the following features are particularly relevant in an AML context [27]:

- **Time, Date**: These temporal features are crucial for chronological analysis, identifying unusual transaction timings (e.g., transactions occurring outside normal business hours for a particular account type), and for engineering time-based features such as transaction velocity or frequency.
- **Sender_account, Receiver_account**: These identify the parties involved in a

transaction. While full network analysis might be an advanced extension, these fields are fundamental for tracking fund flows and identifying patterns like funds being funneled through multiple accounts or concentrating into a single account.

- **Amount**: This is a key indicator for many AML rules, such as those detecting unusually large transactions or patterns of structuring where multiple smaller transactions are made to evade reporting thresholds.
- **Payment_currency, Received_currency**: Discrepancies between payment and received currencies, or transactions conducted in currencies associated with high-risk jurisdictions, can be red flags.
- **Sender_bank_location, Receiver_bank_location**: These geographical features are essential for implementing rules that target transactions involving high-risk jurisdictions, a common AML concern.[10]
- **Payment_type**: Certain types of payments (e.g., 'Cash Deposit', 'Cross-border' transactions as listed in SAML-D features [27]) inherently carry higher AML risks and can be weighted accordingly in risk scoring or rule logic.
- **Is_laundering**: This binary target variable is essential for supervised machine learning approaches or for validating the effectiveness of rule-based detection systems.
- **Laundering_type**: This feature provides more granular labels for different types of suspicious activity, enabling a more nuanced analysis and the potential development of models that can classify the specific nature of the suspected laundering.

It is often the *combination* of these features, rather than individual attributes in isolation, that signals suspicious activity. For example, a single small transaction amount might be perfectly normal. However, a series of small transactions from many different sender accounts converging into a single receiver account (a "fan-in" typology, indicative of layering [31]) would be highly suspicious. This highlights the importance of feature engineering to capture these relational patterns and interactions. Features like Payment_type and Sender/Receiver_bank_location directly facilitate the implementation of risk-based AML approaches, where transactions exhibiting certain characteristics (e.g., a cross-border wire transfer to a high-risk country) are automatically subject to heightened scrutiny.

Table 1: Detailed Feature Breakdown of SAML-D Dataset

Feature Name	Data Type	Description	AML Relevance

Time	object (string)	Time of the transaction	Analyzing transaction timing, identifying off-hours activity, velocity calculations.
Date	object (string)	Date of the transaction	Chronological sequencing, time-window analysis, seasonality, combining with Time for precise timestamp.
Sender_account	int64	Account ID of the sender	Tracking fund origins, identifying networks, velocity/frequency per sender.
Receiver_account	int64	Account ID of the receiver	Tracking fund destinations, identifying networks (e.g., fan-in/fan-out), concentration of funds.
Amount	float64	Value of the transaction	Detecting structuring (below thresholds), unusually large values, velocity of value.
Payment_currency	object (string)	Currency of the payment	Identifying transactions in high-risk currencies, currency mismatches.
Received_currency	object (string)	Currency received by the beneficiary	Identifying currency mismatches with payment currency, unusual currency conversions.

Sender_bank_location	object (string)	Location of the sender's bank	Identifying transactions originating from or involving high-risk jurisdictions.
Receiver_bank_location	object (string)	Location of the receiver's bank	Identifying transactions destined for or involving high-risk jurisdictions.
Payment_type	object (string)	Method of payment (e.g., credit card, cash, ACH, cross-border)	Certain payment types (e.g., cash, cross-border) may carry inherently higher AML risk.
Is_laundering	int64 (0 or 1)	Binary indicator: 1 if suspicious, 0 if normal	Target variable for supervised ML; validation flag for rule-based systems.
Laundering_type	object (string)	Classification of typology (e.g., Normal_Fan_Out, Suspicious_Cycle)	Provides granular insight into the nature of suspicious activity; enables multi-class classification or targeted rule evaluation.

This table provides a consolidated reference for the data fields that will be central to the data exploration, feature engineering, and rule development phases of the project.

IV. Building the AML Data Pipeline: A Multi-Layered Approach

This section details the construction of the end-to-end AML data pipeline, from raw data ingestion through processing and transformation, to advanced analytics and visualization.

A. Overall Data Flow Diagram

The pipeline will follow a logical progression, leveraging the strengths of each

technology at different stages:

1. **Raw Data Ingestion:** The SAML-D dataset (CSV format) is sourced from Kaggle and uploaded to a designated "landing zone" in AWS S3.
2. **Big Data Processing (EMR):** An AWS EMR cluster, configured with Hadoop and Spark, reads the raw CSV data from the S3 landing zone. PySpark scripts are executed on EMR to perform:
    - Data cleansing and preprocessing.
    - Advanced feature engineering tailored for AML detection.
    - Exploratory Data Analysis (EDA) using Spark SQL or Hive. The processed and enriched data is then written back to a "processed" or "curated" zone in AWS S3, preferably in Parquet format for efficiency.
3. **ETL to Data Warehouse (Glue/Spark Connector to Snowflake):** Data from the S3 processed zone is loaded into Snowflake tables. This can be achieved using:
    - An AWS Glue ETL job.
    - The Snowflake Spark Connector (potentially run on the same EMR cluster or a dedicated environment).
4. **AML Analytics in Data Warehouse (Snowflake):**
    - AML detection rules are implemented using Snowflake SQL to query the transaction data and identify suspicious patterns.
    - (Optional Extension) Machine learning models for anomaly detection can be developed and deployed using Snowpark for Python.
    - Transactions flagged as suspicious are stored in a dedicated ALERTS table within Snowflake.
5. **Visualization and Reporting (QuickSight):** AWS QuickSight connects to the Snowflake ALERTS table and other relevant transaction tables to:
    - Create interactive dashboards displaying key AML alerts and risk indicators.
    - Enable visual exploration of suspicious activities.
    - (Optional) Configure threshold-based alerts within QuickSight for specific KPIs.

The organization of data within AWS S3 into distinct zones (e.g., landing, processed, curated) is a recognized data lake best practice.[16] This approach promotes better data management, allows for reprocessing from intermediate stages if necessary, and can support diverse data consumers with varying needs. The choice between AWS Glue and the Snowflake Spark Connector for the ETL step into Snowflake offers flexibility.[43] AWS Glue provides a serverless, managed ETL service, which can simplify job orchestration, while the Spark Connector offers fine-grained control, especially if complex transformations are required during the loading process itself. For this project, AWS Glue might offer a more straightforward path for the initial

implementation.

B. Ingestion & Raw Storage: AWS S3 Data Lake

AWS S3 will serve as the foundational storage layer for the raw and processed data.

1. Setting up S3 Buckets:
   Proper organization of S3 buckets is crucial for a manageable and scalable data lake.42 It is recommended to create a main project bucket with specific prefixes (folders) for different data stages and artifacts:
   - s3://your-aml-project-bucket/landing/: For the raw, unaltered SAML-D CSV file.
   - s3://your-aml-project-bucket/processed/: For data that has been cleaned, transformed, and enriched by EMR (e.g., in Parquet format).
   - s3://your-aml-project-bucket/curated/: (Optional) For data that has undergone further aggregation or structuring specifically for Snowflake ingestion or direct analysis by tools like QuickSight if bypassing Snowflake for certain views.
   - s3://your-aml-project-bucket/logs/: To store logs generated by EMR and AWS Glue jobs, facilitating debugging and monitoring.
   - s3://your-aml-project-bucket/emr-artifacts/: To store PySpark scripts, EMR bootstrap actions, and any other configuration files related to the EMR cluster.
2. Best Practices for S3 Data Lake Organization:
   Beyond the basic structure, adhering to best practices enhances the robustness and maintainability of the S3 data lake:
   - **Consistent Naming Conventions:** Use clear and consistent names for buckets and prefixes.
   - **Versioning:** Enable versioning on S3 buckets, especially for the landing and processed zones, to protect against accidental deletions or overwrites and to allow for rollback if needed.
   - **Security:** Secure buckets using appropriate IAM policies (granting least privilege access to users and services) and S3 bucket policies. Consider encryption at rest.
   - **Data Format:** Storing processed data in a columnar format like Apache Parquet in the processed zone is highly recommended.[16] Parquet offers significant advantages in terms of compression and query performance for analytical workloads in Spark and Snowflake compared to row-oriented formats like CSV. This improves efficiency for downstream processing and analytics.

The S3 data lake functions as the single source of truth for data before it enters

specialized analytical systems like Snowflake. This architectural decoupling allows different tools and services to access the same consistent, processed data, providing flexibility for future enhancements or changes in the analytical stack.

C. Big Data Processing & Transformation: AWS EMR with Hadoop and Spark

AWS EMR (Elastic MapReduce) provides the platform for leveraging the Hadoop ecosystem, particularly Apache Spark, for large-scale data processing and transformation. This stage is critical for preparing the raw transaction data for effective AML analysis.

1. Configuring and Launching an AWS EMR Cluster:
   Setting up an EMR cluster involves several configuration choices 46:
   - **EMR Release:** Select a recent and stable EMR release.
   - **Applications:** Choose the necessary applications, which for this project will primarily be Hadoop, Spark, and optionally Hive (for EDA).
   - **Instance Types and Count:** Select appropriate EC2 instance types for the master, core, and (optional) task nodes. The choice depends on the expected workload size and processing requirements. For development, smaller instance types might suffice, but for demonstrating scalability, using instance types optimized for compute or memory might be considered.
   - **Networking:** Configure EC2 key pairs for SSH access, specify VPC subnets (preferably private subnets for worker nodes with a NAT gateway for internet access if needed), and set up security groups to control network traffic to and from the cluster nodes.
   - **Logging:** Configure EMR to store logs in the designated S3 logs bucket for troubleshooting.
   - **EMRFS (EMR File System):** EMRFS allows the cluster to use S3 as a persistent storage layer, providing consistent view semantics and encryption options.[48] This is the recommended way for EMR to interact with data in S3.
   - **Bootstrap Actions:** Use bootstrap actions to install additional libraries or configure the cluster nodes when they launch (e.g., installing specific Python packages needed by PySpark scripts).
2. Data Ingestion into EMR using PySpark:
   The first step in the EMR processing job is to read the raw SAML-D CSV data from the S3 landing zone into a Spark DataFrame. This can be achieved with a simple PySpark command:
   Python
   ```
   from pyspark.sql import SparkSession

   spark = SparkSession.builder.appName("AML_Data_Processing").getOrCreate()
   ```

```
# Path to the raw SAML-D CSV file in S3
raw_data_path = "s3://your-aml-project-bucket/landing/SAML-D.csv"

# Read the CSV data into a Spark DataFrame
# Ensure header=True to use the first row as column names
# inferSchema=True can be used for initial exploration, but defining schema explicitly is better for production
raw_df = spark.read.csv(raw_data_path, header=True, inferSchema=True)
raw_df.printSchema()
raw_df.show(5)
```

3. Data Cleansing & Preprocessing with PySpark:
   Before feature engineering, the data needs to be cleaned and preprocessed to ensure quality and consistency.26 For the SAML-D dataset:
   - **Handling Missing Values:** Although the SAML-D dataset appears relatively clean based on its description [27], it's good practice to check for and handle any missing values. Strategies could include imputation (filling missing values based on statistical measures) if appropriate for certain fields, or dropping rows/columns if missingness is extensive and imputation is not feasible.
     ```
     Python
     # Example: Check for nulls in each column
     # from pyspark.sql.functions import col, sum as spark_sum
     # raw_df.select([spark_sum(col(c).isNull().cast("int")).alias(c) for c in raw_df.columns]).show()
     ```
   - **Data Type Conversion:** Ensure all columns have the correct data types. For example, Date and Time columns (initially strings) should be converted to proper timestamp or date types. The Amount column should be numeric (e.g., DoubleType or DecimalType). The Kaggle notebook for SAML-D shows an example using Pandas for date conversion [27]; similar transformations can be done in PySpark using functions like to_timestamp or by defining a UDF if custom parsing is needed.
     ```
     Python
     from pyspark.sql.functions import to_timestamp, col, year, month, dayofweek, hour

     # Combine Date and Time into a single transaction_timestamp
     # Assuming 'Date' is like 'YYYY-MM-DD' and 'Time' is like 'HH:MM:SS'
     processed_df = raw_df.withColumn("transaction_timestamp_str", F.concat(F.col("Date"), F.lit(" "), F.col("Time")))
     processed_df = processed_df.withColumn("transaction_timestamp",
     ```

```
to_timestamp("transaction_timestamp_str", "yyyy-MM-dd HH:mm:ss"))
```

- **Initial Transformations:** Create a unified transaction_timestamp field by combining the separate Date and Time columns. This simplifies time-based analysis.

4. Advanced Feature Engineering for AML with PySpark:
   This is a crucial step where domain knowledge about AML is translated into new data features that can help identify suspicious activities.17 These engineered features will significantly enhance the effectiveness of both rule-based systems and machine learning models. Examples relevant to AML include:
   - **Temporal Features:**
     - Extract components from transaction_timestamp like hour of the day, day of the week, month. Transactions occurring at unusual times (e.g., late at night for typical retail accounts) can be indicative of risk.[27]
       Python
       ```
       processed_df = processed_df.withColumn("transaction_hour", hour(col("transaction_timestamp")))
       processed_df = processed_df.withColumn("transaction_day_of_week", dayofweek(col("transaction_timestamp")))
       ```
   - **Transactional Behavior Features (Velocity & Frequency):**
     - Number of transactions per account (Sender_account or Receiver_account) over various rolling time windows (e.g., last 1 hour, 24 hours, 7 days).
     - Total transaction amount per account over the same time windows.
     - Average transaction amount for an account, and the deviation of the current transaction from this average. Significant deviations might be anomalous.[6]
     - These often require window functions in Spark SQL.
       Python
       ```
       from pyspark.sql.window import Window
       from pyspark.sql.functions import count, sum as spark_sum, avg

       # Example: Count of transactions by sender in last 24 hours
       # Define window specification
       # window_spec_24hr = Window.partitionBy("Sender_account") \
       #                         .orderBy(col("transaction_timestamp").cast("long")) \
       #                         .rangeBetween(-24 * 60 * 60, 0) # 24 hours in seconds

       # processed_df = processed_df.withColumn("sender_txn_count_24hr",
       ```

```
count(col("Amount")).over(window_spec_24hr))
```
(Note: True rolling time windows in Spark can be complex and might require careful handling of event time and watermarking if dealing with streaming data. For batch processing on SAML-D, simpler aggregations grouped by account and date components might be a more straightforward starting point for a portfolio project.)

- **Geographical Risk Flags:**
  - Create binary flags if Sender_bank_location or Receiver_bank_location matches a predefined list of (mock) high-risk countries.[11]
    ```
    Python
    high_risk_countries = # Define a list
    processed_df = processed_df.withColumn("is_sender_location_high_risk", col("Sender_bank_location").isin(high_risk_countries))
    processed_df = processed_df.withColumn("is_receiver_location_high_risk", col("Receiver_bank_location").isin(high_risk_countries))
    ```

- **Payment Type Risk Score:**
  - Assign a numerical risk score based on Payment_type. For example, 'Cash Deposit' or 'Cross-border' might receive higher scores than 'Debit Card'.[27] This often involves creating a mapping or using conditional logic.
- **Amount-based Features for Structuring Detection:**
  - Binary flag if Amount is just below common reporting thresholds (e.g., between $9,000 and $9,999 if the threshold is $10,000 [11]).
  - Ratio of transaction amount to a customer's typical balance or transaction size (if such baseline data can be mocked or derived).

The process of feature engineering is often iterative.[17] It involves brainstorming potential indicators, implementing them, and then evaluating their utility in detecting suspicious behavior. Documenting the rationale behind each engineered feature is important.**Table 2: Example Engineered Features for AML Detection**

Feature Name	Description	PySpark Logic/Derivation (Conceptual)	AML Relevance
transaction_hour	Hour of the day the transaction occurred (0-23)	hour(col("transaction_timestamp"))	Detects unusual transaction times (e.g., late night).
sender_daily_txn_count	Number of transactions by sender on that day	Group by Sender_account, date(transaction_tim	High frequency can indicate structuring or rapid movement.

		estamp) and count(*) then join back.	
sender_daily_txn_value	Total value of transactions by sender on that day	Group by Sender_account, date(transaction_timestamp) and sum(Amount) then join back.	Unusually high total value can be suspicious.
is_high_risk_country_txn	Boolean flag if sender or receiver bank location is in a high-risk list	`col("Sender_bank_location").isin(list)	col("Receiver_bank_location").isin(list)`
is_structuring_amount	Boolean flag if amount is just below a known reporting threshold	(col("Amount") > 9000) & (col("Amount") < 10000) & (col("Payment_type") == 'CASH DEPOSIT') (example)	Helps identify potential structuring attempts.
amount_deviation_from_avg	Percentage deviation of current transaction amount from sender's average amount	Calculate sender's historical avg txn amount, then (current_amount - avg_amount) / avg_amount.	Large deviations can signal unusual activity for that customer.

5. Exploratory Data Analysis (EDA) using Spark SQL and Hive on EMR:
   After initial processing and feature engineering, EDA is performed to understand the data characteristics, validate transformations, and refine hypotheses for AML rules.16
   - **Using Hive:** If Hive is included in the EMR cluster, Hive external tables can be created to point to the processed data stored in S3 (in Parquet format). This allows for SQL-based querying of the data.[16]

     SQL
     ```
     -- Example Hive DDL for an external table
     CREATE EXTERNAL TABLE processed_transactions (
       Time STRING,
       Date STRING,
       Sender_account BIGINT,
     ```

```
    --... other original columns
    transaction_timestamp TIMESTAMP,
    transaction_hour INT,
    is_sender_location_high_risk BOOLEAN
    --... other engineered features
)
STORED AS PARQUET
LOCATION 's3://your-aml-project-bucket/processed/your-output-path/';
```

- **Using Spark SQL:** Alternatively, Spark SQL can be used directly on the Spark DataFrames.

```
Python
processed_df.createOrReplaceTempView("processed_transactions_view")
suspicious_payment_types_df = spark.sql("""
    SELECT Payment_type, COUNT(*) as transaction_count, SUM(Amount) as total_amount
    FROM processed_transactions_view
    WHERE Is_laundering = 1
    GROUP BY Payment_type
    ORDER BY total_amount DESC
""")
suspicious_payment_types_df.show()
```

EDA queries might include calculating distributions of transaction amounts, identifying common payment types for suspicious vs. normal transactions, or looking at transaction volumes by bank location.[16] This step helps in understanding the data's nuances before defining more formal detection rules in Snowflake.The EMR processing stage is pivotal for demonstrating proficiency with "Big Data" technologies. While the SAML-D dataset (around 9.5 million rows [27]) might be manageable on a single powerful machine for prototyping, using EMR/Spark showcases the ability to design solutions that can scale to the much larger datasets typically encountered in major financial institutions. The choice to output data from EMR in Parquet format is a critical detail, as this columnar storage significantly enhances the performance of subsequent data loading into Snowflake and any direct analytical queries against the S3 data.[16]

6. Writing Processed Data to S3:
   After all transformations and feature engineering, the resulting DataFrame should be written back to the S3 processed zone in Parquet format.

```
Python
processed_output_path = "s3://your-aml-project-bucket/processed/transactions_enriched/"
processed_df.write.mode("overwrite").parquet(processed_output_path)
```

D. Advanced Analytics & Warehousing: Snowflake

Snowflake serves as the analytical data warehouse where the processed transaction data is loaded, and AML detection logic is applied. Its architecture, which separates storage and compute, is well-suited for the variable and often intensive workloads of AML analytics.[14]

1. ETL to Snowflake: Loading Processed Data from S3:
   The Parquet files from the S3 processed zone need to be loaded into Snowflake tables.
   - **Using AWS Glue:** An AWS Glue ETL job can be configured for this. This involves:
     - Creating a connection to Snowflake in the AWS Glue Data Catalog, securely storing Snowflake credentials (e.g., using AWS Secrets Manager [44]).
     - Authoring a PySpark script within the Glue job that reads the Parquet data from S3 and writes it to the target Snowflake table using the Snowflake Connector for Spark.
     - Relevant parameters for the Snowflake connector include sfUrl, sfUser, sfPassword (or sfRole if using key pair authentication, which is more secure), sfDatabase, sfSchema, and sfWarehouse.[43]
   - **Using Snowflake's COPY INTO command with an External Stage:** This is often a highly efficient method for bulk loading from S3.
     - Create a Storage Integration object in Snowflake to securely connect to S3 without exposing credentials.[52]
     - Create an External Stage object in Snowflake that references the S3 path containing the Parquet files and uses the storage integration.
     - Use the COPY INTO <table> FROM @your_external_stage command to load the data. This command can leverage Snowflake's massively parallel processing (MPP) capabilities for fast ingestion.

```
SQL
-- Example: Assuming storage integration 's3_int' and stage 'aml_processed_stage' are created
-- CREATE OR REPLACE STAGE aml_processed_stage
--   URL = 's3://your-aml-project-bucket/processed/transactions_enriched/'
--   STORAGE_INTEGRATION = s3_int
--   FILE_FORMAT = (TYPE = PARQUET);

-- COPY INTO TRANSACTIONS
--   FROM @aml_processed_stage
--   MATCH_BY_COLUMN_NAME = 'CASE_INSENSITIVE' -- If Parquet schema matches table
```

```
    --   ON_ERROR = 'SKIP_FILE'; -- Or other error handling
```

2. Snowflake Schema Design for AML:
   A well-designed schema in Snowflake is critical for query performance and analytical flexibility.54
   - **Tables:**
     - TRANSACTIONS: This will be the main table storing the processed transaction data from S3, including all original and engineered features.
     - ALERTS: This table will store records of transactions that are flagged as suspicious by either the SQL-based rules or (optionally) machine learning models. It should include references to the original transaction (e.g., transaction ID), the rule/model that triggered the alert, an alert timestamp, and potentially a risk score or alert reason.
     - (Optional) CUSTOMER_PROFILES: A simplified table to mock basic customer information (e.g., CUSTOMER_ID, RISK_TIER, BUSINESS_TYPE, EXPECTED_ACTIVITY_LEVEL). In a real AML system, rich Know Your Customer (KYC) data is extensively used.[10] Even mock data here can help demonstrate more complex rule logic.
   - **Data Types:** Use appropriate Snowflake data types. For instance, TIMESTAMP_NTZ (No Time Zone) or TIMESTAMP_LTZ (Local Time Zone) for transaction timestamps, NUMBER or DECIMAL for monetary amounts, VARCHAR for textual data, and BOOLEAN for flags.[54]
   - **Clustering Keys:** For large tables like TRANSACTIONS, define clustering keys based on frequently queried columns. Common AML queries often filter or join on transaction dates and account identifiers. Therefore, clustering on (TRANSACTION_TIMESTAMP, SENDER_ACCOUNT) or (TRANSACTION_TIMESTAMP, RECEIVER_ACCOUNT) could significantly improve performance.[54] The decision on clustering keys should align with anticipated query patterns.
   - **Materialized Views:** For complex aggregations that are frequently accessed and don't change too often, materialized views can enhance query performance.[57] For example, a materialized view summarizing daily transaction counts and total values per account could speed up certain KRI calculations. However, materialized views in Snowflake have limitations (e.g., can query only a single table, no UDFs, limited aggregate functions [57]), so their applicability must be carefully evaluated.
3. Implementing AML Detection Rules in Snowflake SQL:
   This is where the core AML logic is applied to the transaction data residing in Snowflake. The AML typologies and red flags identified earlier (Section II.C) are

translated into SQL queries.

- **Structuring:** Identify series of transactions just below reporting thresholds.

```
SQL
INSERT INTO ALERTS (TRANSACTION_ID, ALERT_TYPE, ALERT_TIMESTAMP,
RULE_ID, DETAILS)
SELECT
    TRANSACTION_ID,
    'Structuring Attempt',
    CURRENT_TIMESTAMP(),
    'RULE_STRUCT_001',
    'Cash deposit amount ' |
```

| AMOUNT::VARCHAR |
| ' is close to reporting threshold.'
FROM TRANSACTIONS
WHERE AMOUNT > 9000 AND AMOUNT < 10000 -- Assuming $10k threshold
AND PAYMENT_TYPE = 'Cash Deposit' -- Or similar cash-based types
AND IS_LAUNDERING = 0; -- Example: Apply to transactions not already known as laundered
* **Rapid Movement of Funds:** Detect funds deposited and quickly transferred out. This often requires window functions to compare transactions within the same account over short timeframes.sql
-- Conceptual: More complex logic using LAG/LEAD and time differences
-- INSERT INTO ALERTS (...)
-- SELECT... FROM (
-- SELECT TRANSACTION_ID, AMOUNT, TRANSACTION_TIMESTAMP, SENDER_ACCOUNT, RECEIVER_ACCOUNT, PAYMENT_TYPE,
-- LAG(PAYMENT_TYPE) OVER (PARTITION BY SENDER_ACCOUNT ORDER BY TRANSACTION_TIMESTAMP) as prev_payment_type,
-- LAG(AMOUNT) OVER (PARTITION BY SENDER_ACCOUNT ORDER BY TRANSACTION_TIMESTAMP) as prev_amount,
-- TRANSACTION_TIMESTAMP - LAG(TRANSACTION_TIMESTAMP) OVER (PARTITION BY SENDER_ACCOUNT ORDER BY TRANSACTION_TIMESTAMP) as time_diff_from_prev
-- FROM TRANSACTIONS
--)
-- WHERE PAYMENT_TYPE LIKE '%TRANSFER%' AND prev_payment_type LIKE '%DEPOSIT%'
-- AND AMOUNT >= prev_amount * 0.9 -- Moved a significant portion
-- AND EXTRACT(EPOCH FROM time_diff_from_prev) < (24 * 60 * 60); -- Within 24 hours
* **High-Risk Jurisdiction Transactions:** Flag transactions involving predefined high-risk locations.sql
INSERT INTO ALERTS (TRANSACTION_ID, ALERT_TYPE, ALERT_TIMESTAMP, RULE_ID, DETAILS)
SELECT

```
TRANSACTION_ID,
'High-Risk Jurisdiction Transfer',
CURRENT_TIMESTAMP(),
'RULE_GEO_001',
'Transaction involves high-risk location: ' |
| RECEIVER_BANK_LOCATION::VARCHAR
FROM TRANSACTIONS
WHERE IS_RECEIVER_LOCATION_HIGH_RISK = TRUE -- Using engineered feature
AND IS_LAUNDERING = 0;
* **Unusual Transaction Amounts for Customer:** Flag transactions significantly larger than the customer's average.sql
-- Assuming a CUSTOMER_PROFILES table with AVG_TRANSACTION_AMOUNT
-- Or calculate average on the fly if feasible for the project scale
-- INSERT INTO ALERTS (...)
-- SELECT T.TRANSACTION_ID,...
-- FROM TRANSACTIONS T
-- JOIN (SELECT SENDER_ACCOUNT, AVG(AMOUNT) as HISTORICAL_AVG_AMOUNT
-- FROM TRANSACTIONS WHERE TRANSACTION_TIMESTAMP < 'cutoff_date_for_avg_calc' /* to avoid using future data */
-- GROUP BY SENDER_ACCOUNT) C_AVG
-- ON T.SENDER_ACCOUNT = C_AVG.SENDER_ACCOUNT
-- WHERE T.AMOUNT > C_AVG.HISTORICAL_AVG_AMOUNT * 5 -- e.g., 5 times larger
-- AND T.IS_LAUNDERING = 0;
```

The results of these SQL queries (i.e., the flagged transactions and associated alert details) are inserted into the ALERTS table. This table then serves as the primary source for review and visualization.

Table 3: Example AML Detection Rules and SQL Logic in Snowflake

Rule ID	AML Typology Targeted	Description	Snowflake SQL Snippet (Conceptual)	Key Features Used
RULE_STRUCT_001	Structuring	Cash deposits just below $10,000 reporting threshold.	WHERE Amount BETWEEN 9000 AND 9999 AND Payment_type = 'Cash Deposit'	Amount, Payment_type

RULE_GEO_001	High-Risk Jurisdiction	Transactions to/from a country on a high-risk list.	WHERE Receiver_bank_location IN ('HR_Country1', 'HR_Country2') OR Sender_bank_location IN (...) (or use engineered boolean flag)	Receiver_bank_location, Sender_bank_location
RULE_VELOCITY_001	Rapid Movement / Velocity	High number of transactions for an account in a short period (e.g., 24 hours).	GROUP BY Sender_account, DATE(transaction_timestamp) HAVING COUNT(*) > N (N is threshold)	Sender_account, transaction_timestamp
RULE_UNUSUAL_001	Unusual Transaction Pattern	Transaction amount significantly deviates from customer's historical average.	WHERE Amount > (SELECT AVG(Amount) * X FROM... WHERE Account = current_account AND Date < current_date) (X is multiplier)	Amount, Sender_account, transaction_timestamp

4. **(Optional Extension) Leveraging Snowpark for Python for ML-based Anomaly Detection:** Snowpark enables the execution of Python code, including machine learning model training and inference, directly within Snowflake's environment, operating on data stored in Snowflake tables.[15] This significantly reduces data movement and simplifies the operationalization of ML models.
    - **Algorithm Choice:** For anomaly detection in transactions, algorithms like Isolation Forest or K-Means clustering are suitable starting points.
        - **Isolation Forest:** This algorithm is effective at identifying outliers by isolating observations that are "few and different." Anomalous points, by their nature, tend to be further from dense regions of data and thus require fewer random partitions to be isolated in a tree structure.[60]
        - **K-Means Clustering:** While primarily a clustering algorithm, K-Means can

be adapted for anomaly detection by identifying data points that are distant from the centroid of the cluster to which they are assigned.[62] Points with a large distance to their cluster centroid can be considered anomalous.
        - **Principal Component Analysis (PCA):** PCA can be used for dimensionality reduction, and the reconstruction error when projecting data back from a lower-dimensional space can serve as an anomaly score. Transactions with high reconstruction error are less well-represented by the principal components and may be anomalous.[65]
    - **Implementation with Snowpark:**
        - Use Snowpark to create a session and load transaction data from a Snowflake table into a Snowpark DataFrame.
        - Perform any necessary feature scaling or selection within Snowpark.
        - Train a chosen anomaly detection model (e.g., IsolationForest from scikit-learn) using a Snowpark User-Defined Function (UDF) or a Stored Procedure if more complex logic is needed.
        - Score all transactions using the trained model.
        - Insert transactions with anomaly scores exceeding a certain threshold into the ALERTS table, possibly with a different ALERT_TYPE to distinguish them from rule-based alerts. Including this Snowpark component, even as a proof-of-concept, would demonstrate familiarity with cutting-edge Snowflake capabilities and the integration of machine learning into the data warehousing environment, a highly valuable skill.

E. Visualization & Alerting: AWS QuickSight

AWS QuickSight will be used to create dashboards for visualizing AML alerts and Key Risk Indicators (KRIs) derived from the data in Snowflake.

1. **Connecting AWS QuickSight to Snowflake:**
    - Within the AWS QuickSight console, create a new dataset. Select Snowflake as the data source.[67]
    - Provide the necessary Snowflake connection details: Snowflake account identifier, the warehouse to use for queries, the database and schema containing the ALERTS and TRANSACTIONS tables, and user credentials (username/password, or preferably, configure QuickSight to use an IAM role that can assume a role in Snowflake for more secure access if set up).
    - Choose the data access mode:
        - **SPICE (Super-fast, Parallel, In-memory Calculation Engine):** QuickSight imports data into its own optimized in-memory engine. This

typically offers better performance for dashboards with complex visuals or large datasets, and data can be scheduled for refresh.[67] This is likely suitable for the ALERTS table and potentially a sample of the TRANSACTIONS table for this project.
        - **Direct Query:** Queries are run directly against Snowflake each time a visual is loaded or interacted with. This provides real-time data but performance depends on Snowflake warehouse responsiveness and query complexity.[67]
2. Developing a Dashboard for AML Alerts and Key Risk Indicators (KRIs):
   The dashboard should be designed from the perspective of an AML analyst, providing them with the information needed to monitor and investigate suspicious activity.67 Key visuals could include:
    - **Alerts Table:** A detailed table listing current alerts from the Snowflake ALERTS table, showing columns like Transaction ID, Alert Timestamp, Alert Type/Rule ID, Alert Score (if applicable), Amount, Sender/Receiver Information, and key details about why the transaction was flagged.
    - **Key Performance Indicators (KPIs):**
        - Total number of active alerts.
        - Total monetary value of alerted transactions.
        - Number of alerts generated per AML rule or typology.
        - Alerts generated today/this week.
    - **Charts and Graphs:**
        - Alerts over time (e.g., line chart showing daily/weekly alert counts).
        - Distribution of alerted transaction amounts (e.g., histogram).
        - Alerts by payment type (e.g., bar chart).
        - Alerts by sender/receiver bank location (e.g., map visual or bar chart, highlighting high-risk jurisdictions).
    - **Interactivity:** Implement filters (e.g., by date range, alert type, risk score) and consider drill-down capabilities where an analyst can click on an alert to see more detailed transaction information or related account activity.
3. Configuring Threshold-Based Alerts in QuickSight:
   QuickSight allows users to create threshold-based alerts on KPI visuals within a dashboard.69 For example:
    - If a KPI visual displays "Number of Transactions to High-Risk Countries Today," an alert can be set to trigger an email notification if this count exceeds a predefined threshold (e.g., 50 transactions).
    - These alerts are evaluated based on the data refresh schedule of the underlying dataset in QuickSight.[72]

While QuickSight alerts are useful for monitoring dashboard-level metrics, it's worth

noting that for more immediate, operational alerting directly triggered by the detection of suspicious activity in the pipeline (e.g., as soon as a new record is inserted into Snowflake's ALERTS table), other mechanisms might be more suitable in a production environment. These could include Snowflake's native alerting features [73], which can execute an action (like sending an email or calling a webhook) when a SQL condition is met, or AWS services like Lambda functions triggered by database events or S3 notifications, integrated with Amazon SNS for notifications. Mentioning this distinction demonstrates a broader architectural awareness.

V. Step-by-Step Implementation Roadmap

This section provides a sequential, modular guide to building the AML data engineering project. Each module outlines prerequisites, key steps, and expected outcomes.

Module 1: Setting up the AWS Environment

- **Prerequisites:** An active AWS account with permissions to create S3 buckets, IAM roles/policies, EMR clusters, Glue jobs, and QuickSight resources.
- **Steps:**
  1. **IAM Roles & Policies:**
     - Define and create the necessary IAM roles: EMR_Service_Role, EMR_EC2_Instance_Profile_Role, Glue_Job_Role.
     - Create and attach IAM policies to these roles granting least-privilege access to S3 (specific buckets/prefixes for landing, processed, logs, artifacts), AWS Glue Data Catalog (if used by Hive), and AWS Secrets Manager (for Glue to retrieve Snowflake credentials). Refer to Section VI.A for detailed policy structures.
  2. **S3 Buckets:**
     - Create the main S3 bucket for the project (e.g., your-unique-aml-project-cedric).
     - Within this bucket, create the following prefixes (folders): landing/, processed/, curated/ (optional), logs/, emr-artifacts/.
     - Enable versioning and server-side encryption (e.g., SSE-S3) on these buckets/prefixes.
  3. **AWS Secrets Manager (for Snowflake Credentials):**
     - Store Snowflake user credentials (username, password) as a new secret in AWS Secrets Manager. Note the ARN of this secret, as it will be used by the Glue job role.
- **Expected Output:** S3 buckets are created and secured. IAM roles and policies are in place. Snowflake credentials are securely stored.

Module 2: Data Ingestion and Raw Storage

- **Prerequisites:** SAML-D dataset downloaded as a CSV file (e.g., SAML-D.csv). AWS CLI configured or AWS Management Console access.
- **Steps:**
  1. **Upload SAML-D to S3:**
     - Using AWS CLI: aws s3 cp SAML-D.csv s3://your-unique-aml-project-cedric/landing/
     - Or, use the AWS Management Console to upload the file to the landing/ prefix.
- **Expected Output:** SAML-D.csv is present in the s3://your-unique-aml-project-cedric/landing/ location.

Module 3: EMR Cluster Setup and Spark Application Development

- **Prerequisites:** IAM roles (EMR_Service_Role, EMR_EC2_Instance_Profile_Role) configured with necessary S3 and Glue Catalog access. PySpark script for processing developed locally or directly in EMR Notebooks.
- **Steps:**
  1. **Upload PySpark Script to S3:**
     - Place your main PySpark processing script (e.g., aml_spark_job.py) into s3://your-unique-aml-project-cedric/emr-artifacts/scripts/.
  2. **Launch EMR Cluster:**
     - Navigate to the EMR service in the AWS Management Console.
     - Click "Create cluster."
     - **Software Configuration:** Choose a recent EMR release. Select Spark, Hadoop, Hive (optional for EDA).
     - **Hardware:** Select instance types and number of nodes (Master, Core, Task). For development, 1 master and 2 core nodes of a general-purpose type (e.g., m5.xlarge) might be sufficient. Consider Spot instances for Core/Task nodes for cost savings in non-critical runs.
     - **General Cluster Settings:** Set a cluster name. Configure logging to s3://your-unique-aml-project-cedric/logs/emr/.
     - **Security:** Select your EC2 key pair for SSH. Choose the EMR_Service_Role and EMR_EC2_Instance_Profile_Role. Configure VPC and subnets.
     - **Steps (Optional at Launch):** You can add the Spark application as a step during cluster creation or submit it later.
     - **Bootstrap Actions (Optional):** If your PySpark script requires specific Python libraries not included by default, add a bootstrap action to install them (e.g., sudo python3 -m pip install <library>).

    - Launch the cluster. Wait for it to reach the "Waiting" state.
3. **Develop/Test PySpark Script (if not done):**
    - The script should:
        - Initialize a SparkSession.
        - Read the raw CSV from s3://.../landing/SAML-D.csv.
        - Perform data cleansing (handle nulls, convert data types).
        - Implement feature engineering as outlined in Section IV.C.4 and Table 2.
        - (Optional) Perform EDA using Spark SQL and save/print results.
        - Write the processed DataFrame to s3://.../processed/transactions_enriched/ in Parquet format, using mode("overwrite").
4. **Submit Spark Application to EMR (if not added as a step at launch):**
    - Go to your EMR cluster's "Steps" tab.
    - Add a new step:
        - **Step type:** Spark application.
        - **Application location:** s3://your-unique-aml-project-cedric/emr-artifacts/scripts/aml_spark_job.py
        - **Spark-submit options:** (If needed, e.g., --deploy-mode cluster)
        - **Arguments:** (If your script takes arguments)
    - Add the step and monitor its progress. Check logs in S3 or the EMR console for errors.

- **Expected Output:** EMR job completes successfully. Processed data (Parquet files) is present in s3://your-unique-aml-project-cedric/processed/transactions_enriched/. Logs are available in the S3 logs bucket.

Module 4: Snowflake Setup and Data Loading

- **Prerequisites:** Active Snowflake account with a user having ACCOUNTADMIN or equivalent privileges to create databases, schemas, tables, stages, and storage integrations. Processed data available in S3 (from Module 3).
- **Steps:**
    1. **Snowflake Worksheet Setup:** Connect to your Snowflake account using Snowsight or SnowSQL.
    2. **Create Database, Schema, and Warehouse:**
    ```
    SQL
    CREATE DATABASE IF NOT EXISTS AML_PROJECT_DB;
    CREATE SCHEMA IF NOT EXISTS AML_PROJECT_DB.AML_SCHEMA;
    ```

```
CREATE WAREHOUSE IF NOT EXISTS AML_WH WITH
  WAREHOUSE_SIZE = 'X-SMALL'
  AUTO_SUSPEND = 60 -- Suspend after 1 min of inactivity
  AUTO_RESUME = TRUE;

USE DATABASE AML_PROJECT_DB;
USE SCHEMA AML_SCHEMA;
USE WAREHOUSE AML_WH;
```

3. **Create Snowflake Storage Integration for S3:**
   - This allows Snowflake to securely access your S3 bucket without explicit credentials.

     ```
     SQL
     CREATE OR REPLACE STORAGE INTEGRATION s3_aml_integration
      TYPE = EXTERNAL_STAGE
      STORAGE_PROVIDER = S3
      ENABLED = TRUE
      STORAGE_AWS_ROLE_ARN =
     'arn:aws:iam::YOUR_AWS_ACCOUNT_ID:role/Snowflake_S3_Access_Role' -- Role to be
     created in AWS
      STORAGE_ALLOWED_LOCATIONS =
     ('s3://your-unique-aml-project-cedric/processed/',
     's3://your-unique-aml-project-cedric/landing/');
       -- Optional: STORAGE_BLOCKED_LOCATIONS
     ```

   - After creating the integration, describe it to get the STORAGE_AWS_IAM_USER_ARN and STORAGE_AWS_EXTERNAL_ID:

     ```
     SQL
     DESC INTEGRATION s3_aml_integration;
     ```

   - In AWS IAM, create the role specified in STORAGE_AWS_ROLE_ARN (e.g., Snowflake_S3_Access_Role).
     - **Trusted entity type:** Another AWS account.
     - **Account ID:** The AWS account ID where Snowflake is hosted (Snowflake provides this, or it can be found). For this step, you'll use the STORAGE_AWS_IAM_USER_ARN from the DESC INTEGRATION output to establish the trust relationship. The principal will be the STORAGE_AWS_IAM_USER_ARN.
     - **External ID:** Use the STORAGE_AWS_EXTERNAL_ID from the DESC INTEGRATION output.
     - **Permissions:** Attach a policy granting read access to

s3://your-unique-aml-project-cedric/processed/* and s3://your-unique-aml-project-cedric/landing/*.

4. **Create File Format and External Stage in Snowflake:**

```
SQL
CREATE OR REPLACE FILE FORMAT parquet_format TYPE = PARQUET;

CREATE OR REPLACE STAGE aml_processed_s3_stage
  URL = 's3://your-unique-aml-project-cedric/processed/transactions_enriched/'
  STORAGE_INTEGRATION = s3_aml_integration
  FILE_FORMAT = parquet_format;
```

   - Verify the stage by listing files: LIST @aml_processed_s3_stage;

5. **Create Target Tables in Snowflake:**

```
SQL
-- TRANSACTIONS Table (adjust columns based on your EMR output schema)
CREATE OR REPLACE TABLE TRANSACTIONS (
    Time STRING,
    Date_str STRING, -- Assuming original date as string
    Sender_account BIGINT,
    Receiver_account BIGINT,
    Amount NUMBER(38,2),
    Payment_currency STRING,
    Received_currency STRING,
    Sender_bank_location STRING,
    Receiver_bank_location STRING,
    Payment_type STRING,
    Is_laundering INT, -- 0 or 1
    Laundering_type STRING,
    -- Engineered Features
    TRANSACTION_ID STRING, -- Assuming you generate a unique ID in Spark
    TRANSACTION_TIMESTAMP TIMESTAMP_NTZ,
    TRANSACTION_HOUR INT,
    TRANSACTION_DAY_OF_WEEK INT,
    IS_SENDER_LOCATION_HIGH_RISK BOOLEAN,
    IS_RECEIVER_LOCATION_HIGH_RISK BOOLEAN,
    -- Add other engineered features from your Spark job
    PRIMARY KEY (TRANSACTION_ID) -- If TRANSACTION_ID is unique
);
-- Clustering Key (example)
-- ALTER TABLE TRANSACTIONS CLUSTER BY (TRANSACTION_TIMESTAMP, Sender_account);
```

```
-- ALERTS Table
CREATE OR REPLACE TABLE ALERTS (
    ALERT_ID STRING DEFAULT UUID_STRING(), -- Auto-generated unique ID
    TRANSACTION_ID STRING,
    ALERT_TIMESTAMP TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP(),
    ALERT_TYPE VARCHAR(255), -- e.g., 'Structuring', 'High-Risk Geo', 'ML Anomaly'
    RULE_ID VARCHAR(50), -- Identifier for the rule that triggered the alert
    RISK_SCORE NUMBER(5,2), -- Optional, for ML-based alerts
    DETAILS VARCHAR,
    STATUS VARCHAR(50) DEFAULT 'NEW', -- e.g., NEW, UNDER_REVIEW, CLOSED
    FOREIGN KEY (TRANSACTION_ID) REFERENCES
TRANSACTIONS(TRANSACTION_ID) -- If PK is set on TRANSACTIONS
);
```

6. **Load Data using COPY INTO:**
   ```
   SQL
   COPY INTO TRANSACTIONS
    FROM @aml_processed_s3_stage
    MATCH_BY_COLUMN_NAME = CASE_SENSITIVE -- or CASE_INSENSITIVE based on
   your Parquet field names
    ON_ERROR = 'SKIP_FILE_5%'; -- Example error handling
   ```
   - Verify data load: SELECT COUNT(*) FROM TRANSACTIONS; SELECT * FROM TRANSACTIONS LIMIT 10;

- **Expected Output:** Snowflake database, schema, warehouse, tables, stage, and integration are created. Processed transaction data is loaded into the TRANSACTIONS table. The ALERTS table is empty initially.

Module 5: Implementing AML Analytics in Snowflake

- **Prerequisites:** TRANSACTIONS table populated in Snowflake.
- **Steps:**
  1. **Develop and Test AML SQL Rules:**
     - Write SQL queries based on the rules defined in Section IV.D.3 and Table 3.
     - Test each rule individually to ensure it correctly identifies target transactions.
     - Example:
       ```
       SQL
       -- Test query for structuring
       ```

```
SELECT TRANSACTION_ID, Amount, Payment_type, Sender_bank_location
FROM TRANSACTIONS
WHERE Amount > 9000 AND Amount < 9999.99 AND Payment_type = 'Cash
Deposit';
```

2. **Populate ALERTS Table:**
   - Use INSERT INTO ALERTS SELECT... statements to populate the ALERTS table with transactions flagged by your rules.

```
SQL
-- Example: Inserting alerts for structuring
INSERT INTO ALERTS (TRANSACTION_ID, ALERT_TYPE, RULE_ID, DETAILS)
SELECT
    TRANSACTION_ID,
    'Structuring Attempt',
    'RULE_STRUCT_001',
    'Cash deposit amount ' |
```

```
| AMOUNT::VARCHAR |
| ' is close to $10k reporting threshold.'
FROM TRANSACTIONS
WHERE Amount > 9000 AND Amount < 9999.99
AND Payment_type = 'Cash Deposit';
```

```
    -- Add similar INSERT statements for other rules
    ```
```

3. **(Optional Extension) Snowpark for ML Anomaly Detection:**
   * If pursuing this, develop Python scripts using the Snowpark library.
   * Create Snowpark UDFs or Stored Procedures to train and apply an anomaly detection model (e.g., Isolation Forest) on the `TRANSACTIONS` data.
   * Insert transactions flagged by the ML model into the `ALERTS` table, with an appropriate `ALERT_TYPE` (e.g., 'ML_Anomaly_IsolationForest') and `RISK_SCORE`.

- **Expected Output:** The ALERTS table is populated with suspicious transactions identified by the SQL rules (and optionally ML models).

**Module 6: Visualization and Alerting with QuickSight**

- **Prerequisites:** ALERTS and TRANSACTIONS tables populated in Snowflake. AWS
```

QuickSight Enterprise Edition account.

- **Steps:**
  1. **Connect QuickSight to Snowflake:**
     - In QuickSight, click "Manage data" -> "New dataset."
     - Select "Snowflake" as the data source.
     - Enter connection details: Name for the data source (e.g., Snowflake_AML_Project), Snowflake account identifier, username/password (or use IAM role if configured), warehouse (AML_WH), database (AML_PROJECT_DB), schema (AML_SCHEMA).
     - Validate the connection.
  2. **Create Datasets in QuickSight:**
     - Create one dataset from the ALERTS table.
     - Create another dataset from the TRANSACTIONS table (or a sample/view if it's very large for dashboarding performance). Consider using SPICE for these datasets for better dashboard responsiveness.
     - Define any necessary joins between these datasets within QuickSight if you plan to display related transaction details directly with alerts.
  3. **Build the AML Dashboard:**
     - Create a new Analysis in QuickSight.
     - Add visuals (KPIs, tables, charts) as outlined in Section IV.E.2, using the datasets created.
     - Example visuals:
       - KPI: Count of Alert_ID from the ALERTS dataset.
       - Table: Displaying columns from the ALERTS dataset.
       - Bar Chart: Count of Alert_ID by ALERT_TYPE.
     - Arrange visuals logically on the dashboard. Add titles and descriptions.
  4. **Publish Dashboard:** Once satisfied, publish the analysis as a Dashboard.
  5. **(Optional) Configure QuickSight Threshold Alerts:**
     - On a KPI visual in the published dashboard (e.g., "Total New Alerts Today"), use the visual's menu to "Add alert."
     - Define the threshold and notification preferences (e.g., email).
- **Expected Output:** An interactive QuickSight dashboard visualizing AML alerts and KRIs. (Optional) Email notifications for QuickSight threshold alerts are configured.

Modular testing is key. After each module, verify the outputs (e.g., files in S3, tables in Snowflake, dashboard visuals) before proceeding. Parameterize scripts for S3 paths, database names, etc., to make the project more adaptable.

VI. Security & Operational Best Practices

Implementing robust security and adhering to operational best practices are non-negotiable in financial systems, even for a portfolio project using synthetic data. Demonstrating an understanding of these principles adds significant value.

A. IAM Configuration: Roles and Policies

The principle of least privilege should govern all IAM configurations. This means granting only the necessary permissions required for each component to perform its specific tasks.

- **EMR_Service_Role (Service Role for EMR):** This role is assumed by the Amazon EMR service itself to provision and manage cluster resources (EC2 instances, EBS volumes, etc.) and interact with other AWS services on your behalf (e.g., EC2, S3 for logging).[74] AWS provides a managed policy AmazonEMRServicePolicy_v2_AWS_Managed that can be used, or a custom policy can be crafted if more granular control is needed.
- **EMR_EC2_DefaultRole (Instance Profile for EMR EC2 instances):** This role is attached to the EC2 instances within the EMR cluster, granting them permissions to perform tasks. For this project, it needs:
  - **S3 Access:**
    - Read access (s3:GetObject*, s3:ListBucket) to the landing zone (e.g., arn:aws:s3:::your-aml-project-bucket/landing/*).
    - Write access (s3:PutObject*, s3:DeleteObject*) to the processed zone (e.g., arn:aws:s3:::your-aml-project-bucket/processed/*), the logs zone, and the emr-artifacts zone. .[74]
  - **AWS Glue Data Catalog Access (if Hive metastore is integrated with Glue):** Permissions like glue:GetDatabase, glue:GetTable, glue:GetPartitions, glue:CreatePartition on the relevant Glue Catalog resources.[74]
  - **Example Policy Snippet for EMR_EC2_DefaultRole (S3 & basic Glue access):**

```JSON
{
    "Version": "2012-10-17",
    "Statement":,
            "Resource": [
                "arn:aws:s3:::your-aml-project-bucket/landing/*",
                "arn:aws:s3:::your-aml-project-bucket/landing"
            ]
        },
```

```
        {
            "Sid": "S3ProcessedLogsArtifactsWrite",
            "Effect": "Allow",
            "Action":,
            "Resource": [
                "arn:aws:s3:::your-aml-project-bucket/processed/*",
                "arn:aws:s3:::your-aml-project-bucket/processed",
                "arn:aws:s3:::your-aml-project-bucket/logs/emr/*",
                "arn:aws:s3:::your-aml-project-bucket/logs/emr",
                "arn:aws:s3:::your-aml-project-bucket/emr-artifacts/*",
                "arn:aws:s3:::your-aml-project-bucket/emr-artifacts"
            ]
        },
        {
            "Sid": "GlueCatalogAccessForHive",
            "Effect": "Allow",
            "Action":,
            "Resource":
        }
    ]
}
```

- **Glue_Job_Role:** This role is assumed by AWS Glue ETL jobs. It requires:
    - Read access to the S3 processed zone (e.g., arn:aws:s3:::your-aml-project-bucket/processed/*).
    - Permissions to write to Snowflake. This is typically managed by the Snowflake connector using credentials; the Glue role itself might need network permissions (e.g., VPC access if Snowflake is in a private VPC) and permissions to write to a temporary S3 staging location if the connector uses COPY commands internally.
    - Read access to AWS Secrets Manager to retrieve Snowflake credentials if stored there.[44]
    - Permissions to write logs to CloudWatch Logs and S3 (usually part of the default AWSGlueServiceRole policy).
    - **Example Policy Snippet for Glue_Job_Role (S3 read & Secrets Manager access):**

      JSON
      ```
      {
          "Version": "2012-10-17",
      ```

```
    "Statement":,
            "Resource": [
                "arn:aws:s3:::your-aml-project-bucket/processed/*",
                "arn:aws:s3:::your-aml-project-bucket/processed"
            ]
        },
        {
            "Sid": "SecretsManagerReadForSnowflakeCreds",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource":
"arn:aws:secretsmanager:YOUR_REGION:YOUR_ACCOUNT_ID:secret:your_snowflake_secret_n
ame-???????"
        },
        {
            "Sid": "GlueServiceRolePermissions", // Basic permissions needed by Glue
            "Effect": "Allow",
            "Action":,
            "Resource": "*" // Scope down if possible
        },
        {
            "Sid": "CloudWatchLogsWrite",
            "Effect": "Allow",
            "Action":,
            "Resource": "arn:aws:logs:*:*:*:/aws-glue/*"
        },
        {
            "Sid": "S3WriteForGlueScriptsAndTemp", // For Glue's own script/temp storage
            "Effect": "Allow",
            "Action": ["s3:PutObject"],
            "Resource": [
                "arn:aws:s3:::aws-glue-assets-*/glue-connections/*", // If using Glue connections
                "arn:aws:s3:::aws-glue-temporary-*/*" // Glue temp directories
            ]
        }
    ]
}
```

- **Snowflake Storage Integration IAM Role (e.g., Snowflake_S3_Access_Role):**

This is an IAM role created in the AWS account that Snowflake will assume to access S3 buckets directly (e.g., for COPY INTO commands using an external stage, or for Snowpipe).

- **Permissions Policy:** This policy, attached to the Snowflake_S3_Access_Role, grants necessary S3 permissions (e.g., s3:GetObject, s3:ListBucket for reading from s3://your-aml-project-bucket/processed/, and potentially s3:PutObject if Snowflake needs to write back to S3).[52]
- **Trust Policy:** The crucial part is the trust relationship. This policy on the Snowflake_S3_Access_Role allows the IAM user created by Snowflake (obtained from DESC STORAGE INTEGRATION) to assume this role, using an external ID for added security.[52]

Securely managing how AWS services interact with each other and with Snowflake is a critical aspect of cloud data architecture. Using IAM roles and storage integrations, rather than hardcoding access keys, is a fundamental security best practice.[44] Storing sensitive credentials like Snowflake passwords in AWS Secrets Manager and granting the Glue job role permission to retrieve them is the recommended approach.[44]

Table 4: IAM Role Permissions Summary (Conceptual)

IAM Role	AWS Service Accessed	Key Permissions Granted (Examples)	Resource ARN(s) (Examples)	Rationale/Purpose
EMR_Service_Role	EC2, S3 (for logs), IAM, etc.	ec2:RunInstances, s3:PutObject (for logs), iam:PassRole	Varies, often AWS managed policies like AmazonEMRServicePolicy_v2_AWS_Managed	Allows EMR service to provision and manage cluster resources.
EMR_EC2_DefaultRole	S3, AWS Glue Data Catalog (optional)	s3:GetObject (landing), s3:PutObject (processed, logs), glue:GetTable, glue:GetPartitions	arn:aws:s3:::your-bucket/landing/*, arn:aws:s3:::your-bucket/processed/*, arn:aws:glue:region:acct-id:table/db_name/*	Allows EMR cluster instances to read raw data, write processed data, and interact with Hive metastore (if Glue).

Glue_Job_Role	S3, AWS Secrets Manager, (Network for Snowflake)	s3:GetObject (processed), secretsmanager :GetSecretValue	arn:aws:s3:::your-bucket/processed/*, arn:aws:secretsmanager:region:acct-id:secret:snowflake_creds-*	Allows Glue ETL jobs to read processed data from S3 and securely fetch Snowflake credentials.
Snowflake_S3_Access_Role (AWS)	S3	s3:GetObject, s3:ListBucket (for processed zone)	arn:aws:s3:::your-bucket/processed/*	Role assumed by Snowflake (via Storage Integration) to directly access S3 for COPY commands or external tables.

B. Data Encryption Strategies

Data encryption is vital for protecting sensitive information, even when dealing with synthetic datasets in a development environment, as it establishes good security hygiene.

- **Encryption at Rest:**
  - **AWS S3:** Enable server-side encryption for all S3 buckets used in the project. Options include SSE-S3 (S3-managed keys) or SSE-KMS (AWS Key Management Service-managed keys) for more control.[48]
  - **Snowflake:** Snowflake automatically encrypts all data stored within it (data at rest) using strong AES-256 encryption. This is a managed feature and requires no explicit configuration by the user for standard tables.[14]
- **Encryption in Transit:**
  - Ensure all communication with AWS service endpoints (S3, EMR, Glue, QuickSight) uses HTTPS/TLS. This is generally the default.
  - Connections to Snowflake (e.g., via JDBC/ODBC from Glue, Spark, or QuickSight) should also use TLS encryption, which is standard for Snowflake connectors.

Understanding the shared responsibility model in the cloud is important here. While AWS and Snowflake provide robust encryption capabilities, the user is responsible for correctly configuring and utilizing these features, such as enabling SSE on S3

buckets.[48]

C. Cost Optimization Considerations

Managing costs effectively is a key aspect of working with cloud services.

- **AWS EMR:**
  - **Instance Selection:** Choose instance types appropriate for the workload. For development, smaller instances may suffice.
  - **Spot Instances:** Utilize EC2 Spot Instances for EMR core or task nodes, especially for fault-tolerant workloads, as they can offer significant cost savings compared to On-Demand instances.[48]
  - **Auto-Termination:** Configure EMR clusters to auto-terminate after a period of inactivity to avoid paying for idle resources.
- **AWS S3:**
  - **Storage Classes:** While standard S3 storage is suitable for actively used data in this project, for long-term archival of logs or older processed data in a production scenario, consider S3 Lifecycle policies to transition data to more cost-effective storage classes like S3 Glacier.
- **Snowflake:**
  - **Warehouse Management:** Snowflake's separation of storage and compute allows for granular cost control. Suspend virtual warehouses when they are not in use to avoid compute charges. Warehouses can be configured to auto-suspend after a specified period of inactivity.[14]
  - **Warehouse Sizing:** Choose the appropriate warehouse size (X-Small, Small, etc.) for the workload. Start small and scale up if necessary.
  - **Query Optimization:** Efficiently written SQL queries consume fewer compute resources and thus reduce costs. Proper schema design, including the use of clustering keys [54], contributes to query performance and cost-effectiveness.[55]
- **AWS Glue:**
  - Monitor Data Processing Unit (DPU) usage. Optimize ETL scripts to be efficient. AWS Glue is serverless, so payment is for the resources consumed during job execution.[42]

The serverless nature of services like AWS Glue and Snowflake's pay-per-use compute model are inherently cost-effective if managed correctly.[14] This project provides an opportunity to demonstrate an understanding of these cloud cost management principles.

VII. Crafting Your Portfolio Piece & Future Directions

Successfully completing this project provides a strong foundation for a compelling portfolio piece that can be shared with prospective employers or your new team at Citi.

A. Structuring Your Project for a Compelling Portfolio

To maximize the impact of this project, consider the following:

1. **GitHub Repository:**
    - Create a public GitHub repository to host all project artifacts.
    - Include all PySpark scripts (.py), Snowflake SQL DDL and DML scripts (.sql), example IAM policy JSON templates, and any configuration files.
    - The cornerstone of the repository should be a detailed README.md file.
2. **Comprehensive README.md:**
    - **Project Overview:** Briefly describe the project's purpose (AML transaction monitoring), the problem it addresses, and the technologies used.
    - **Architecture:** Include the conceptual architecture diagram (from Section II.A or IV.A) and explain the data flow through the different components (S3, EMR, Glue, Snowflake, QuickSight).
    - **Setup Instructions:** Provide clear, step-by-step instructions on how to set up the environment and run the project. This should cover:
        - Prerequisites (AWS account, Snowflake account, software to install locally if any).
        - S3 bucket creation and data upload.
        - IAM role and policy setup (referencing the JSON templates).
        - EMR cluster configuration and Spark job submission.
        - Snowflake object creation (database, schema, tables, stages, integration).
        - AWS Glue job setup (if used for ETL to Snowflake).
        - QuickSight connection to Snowflake and dashboard import/recreation.
    - **Code Explanation:** Briefly explain the logic of key PySpark transformations and Snowflake SQL rules.
    - **Demonstration:** Outline steps to demonstrate the pipeline end-to-end, showing how raw data is processed and how alerts are generated and visualized.
    - **Skills Demonstrated:** Explicitly list the skills and technologies utilized and how they were applied in the project (e.g., "Distributed data processing of X million records using PySpark on AWS EMR," "Snowflake schema design with clustering keys for AML analytics," "End-to-end data pipeline orchestration in AWS").
3. **Documentation:** Ensure code is well-commented. The README.md should be

the primary source of documentation.
4. **Demo Script/Screencast (Optional but Recommended):** A short screencast or a well-documented script that walks through running the pipeline and showcasing the final dashboard can be very effective.

The portfolio piece should narrate a story: "An AML challenge (detecting suspicious transactions) was identified. A modern data engineering solution was designed and implemented using industry-relevant technologies (Hadoop/Spark, AWS, Snowflake). This solution successfully processed data and generated actionable (simulated) insights." This narrative structure makes the project more engaging and highlights problem-solving abilities. Clearly articulating the specific skills learned and demonstrated with each technology is crucial for making a strong impression.

B. Potential Enhancements (Future Directions)

Demonstrating an understanding of how the project could be extended shows foresight and a deeper grasp of the AML domain and data engineering possibilities. Potential future enhancements include:

1. **Real-time Data Ingestion & Processing:**
   - Integrate Apache Kafka for streaming raw transaction data into the pipeline.[12]
   - Use Spark Streaming or other stream processing engines (like Amazon Kinesis Data Analytics) on EMR or as a separate service to perform near real-time feature engineering and rule application.
2. **Advanced Machine Learning Models:**
   - Explore more sophisticated ML models for anomaly detection within Snowpark or by integrating with Amazon SageMaker.[7] Examples include:
     - Graph Neural Networks (GNNs) for analyzing transaction networks to uncover complex laundering rings.[6]
     - Deep learning models like autoencoders for unsupervised anomaly detection on high-dimensional transaction data.[51]
3. **Graph Analytics:**
   - Utilize graph databases (e.g., Amazon Neptune) or graph processing libraries (like Spark GraphX) to model relationships between accounts, individuals, and other entities. This can reveal complex patterns like circular fund movements or hidden beneficial ownership structures that are difficult to detect with tabular analysis alone.[6]
4. **Case Management Integration (Conceptual):**
   - Discuss how the alerts generated by the pipeline could theoretically feed into an AML case management system. This system would allow analysts to investigate alerts, document findings, and manage Suspicious Activity Report

(SAR) filings.

5. **Enhanced Data Quality Monitoring & Governance:**
    - Implement automated data quality checks at various stages of the pipeline (e.g., using Deequ on Spark or custom SQL checks in Snowflake).
    - Integrate data lineage tracking tools to provide a clear audit trail of data transformations.
6. **Incorporation of External Data Sources:**
    - Expand the pipeline to ingest and correlate transaction data with external risk indicators, such as:
        - Sanctions lists and Politically Exposed Persons (PEP) watchlists.[10]
        - Adverse media feeds (news articles, social media related to financial crime).[10]
        - Geographic risk data from sources like the FATF or Transparency International.[10]
7. **Automated Model Retraining and Rule Tuning:**
    - Develop processes for periodically retraining ML models with new data and for evaluating and tuning AML detection rules based on feedback from (hypothetical) investigations and emerging typologies. Generative AI could even assist in suggesting rule refinements.[17]

Many of these enhancements align with the industry's push towards more proactive, intelligent, and adaptive AML systems capable of countering the ever-evolving tactics of financial criminals. They also represent significant data engineering challenges, offering further avenues for skill development and specialization.

VIII. Conclusion: Demonstrating Your AML Data Engineering Prowess

This project, centered on building an AML suspicious activity detection system using the Hadoop ecosystem, AWS Cloud, and Snowflake, offers a robust platform to develop and showcase a range of highly sought-after data engineering skills.

A. Summary of Skills and Technologies Mastered

Upon completion, this project will demonstrate proficiency and practical experience in:

- **Big Data Technologies:**
    - **Apache Hadoop & HDFS:** Understanding of distributed storage concepts.
    - **Apache Spark (PySpark):** Large-scale data processing, distributed ETL, complex transformations, feature engineering, and SQL-based analytics on

EMR.
    - **Apache Hive (Optional):** Data warehousing concepts on Hadoop, SQL-based EDA.
- **AWS Cloud Services:**
    - **Amazon S3:** Data lake implementation, storage tiering concepts, security.
    - **Amazon EMR:** Cluster configuration, management, job submission, and integration with S3.
    - **AWS Glue:** Serverless ETL, Data Catalog integration, connecting to Snowflake.
    - **AWS IAM:** Configuring roles and policies for secure, least-privilege access across services.
    - **AWS Secrets Manager:** Secure credential management.
    - **AWS QuickSight:** Data visualization, dashboard creation, connecting to cloud data warehouses.
- **Snowflake Data Cloud:**
    - Cloud data warehousing concepts, schema design, data loading from S3.
    - Advanced SQL for implementing complex AML detection rules.
    - Performance optimization techniques (e.g., clustering keys).
    - (Optional) Snowpark for Python for in-database machine learning and advanced analytics.
- **Data Engineering Core Competencies:**
    - End-to-end data pipeline design and implementation.
    - ETL/ELT processes.
    - Data modeling for analytical systems.
    - Data cleansing and preparation.
    - Advanced feature engineering tailored to a specific domain (AML).
    - Data security and governance best practices in a cloud environment.
    - Cost optimization strategies for cloud data services.
- **AML Domain Understanding:**
    - Knowledge of money laundering stages and common typologies.
    - Ability to translate AML requirements into technical solutions and detection logic.

B. The Value of Such a Project to an AML Team at Citi

Presenting this project to the AML team at Citi will highlight several valuable attributes:

1. **Proactive Learning and Initiative:** Undertaking a complex project based on the manager's technological recommendations demonstrates a strong desire to

learn, adapt quickly, and prepare effectively for the role.

2. **Technical Aptitude with Relevant Technologies:** The project directly showcases hands-on experience with the specific technology stack (Hadoop, AWS, Snowflake) that the team utilizes or is strategically important to the organization.
3. **End-to-End System Building Capability:** Successfully designing and implementing a multi-component pipeline from data ingestion to visualization demonstrates an ability to think architecturally and solve integration challenges—skills crucial in large enterprise environments.
4. **Understanding of AML Principles:** The focus on AML typologies and the development of detection logic (rules or ML models) shows an appreciation for the business context and the goals of an AML team, moving beyond purely technical execution.
5. **Problem-Solving Skills:** Navigating the complexities of setting up cloud services, writing distributed processing jobs, designing database schemas, and integrating different platforms inherently involves problem-solving, a core engineering competency.
6. **Preparedness for Real-World Challenges:** The project simulates many of the tasks and challenges an AML data engineer would face, such as dealing with large datasets (conceptually, via scalable tools), ensuring data quality, implementing detection logic, and making data accessible for analysis.

By successfully completing and articulating the value of this project, an aspiring data engineer can make a significant positive impression, demonstrating not only technical proficiency but also a proactive and solution-oriented mindset that is invaluable to any AML technology team. This endeavor serves as a powerful testament to one's readiness to contribute meaningfully from the outset of employment.

Works cited

1. The 3 Stages of Money Laundering: Placement, Layering, & Integration - Blog | Unit21, accessed June 3, 2025, https://www.unit21.ai/blog/aml-stages
2. TYPOLOGIES AND INDICATORS OF MONEY LAUNDERING 2024, accessed June 3, 2025, https://fid.gov.lv/uploads/files/2023/Typologies%20nad%20Indicators%20of%20ML_3rd%20edition.pdf
3. Money Laundering Using New Payment Methods - FATF, accessed June 3, 2025, https://www.fatf-gafi.org/en/publications/Methodsandtrends/Moneylaunderingusingnewpaymentmethods.html
4. Money Laundering Using New Payment Methods - FATF, accessed June 3, 2025, https://www.fatf-gafi.org/content/dam/fatf-gafi/reports/ML%20using%20New%2

0Payment%20Methods.pdf
5. Top Data Science Projects for Real-World Impact, accessed June 3, 2025, https://www.dasca.org/world-of-data-science/article/top-data-science-projects-for-real-world-impact
6. The Role of Data Science in Combating Money Laundering, accessed June 3, 2025, https://opendatascience.com/the-role-of-data-science-in-combating-money-laundering/
7. 7 Real-World Business Intelligence Automation Examples - RTS Labs, accessed June 3, 2025, https://rtslabs.com/business-intelligence-automation
8. Scalable, actionable anti-money laundering infrastructure from Red Hat - Contentstack, accessed June 3, 2025, https://eu-assets.contentstack.com/v3/assets/blt7dacf616844cf077/blt4fa78e76a92639e9/679b89579724e232552af7b3/Scalable-actionable-anti-money.pdf
9. AML AI overview | Anti Money Laundering AI - Google Cloud, accessed June 3, 2025, https://cloud.google.com/financial-services/anti-money-laundering/docs/concepts/overview
10. The essential guide to AML data: 9 key data types that drive smarter decision-making, accessed June 3, 2025, https://complyadvantage.com/insights/aml-data-types-every-solution-must-offer/
11. AML Transaction Monitoring Rules: Best Examples (2025) - Sumsub, accessed June 3, 2025, https://sumsub.com/blog/aml-transaction-monitoring-rules-scenarios/
12. Hadoop Ecosystem - Simplilearn.com, accessed June 3, 2025, https://www.simplilearn.com/tutorials/hadoop-tutorial/hadoop-ecosystem
13. Engineering Sustainable Data Architectures for Modern Financial ..., accessed June 3, 2025, https://www.mdpi.com/2079-9292/14/8/1650
14. Key Concepts & Architecture | Snowflake Documentation, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/intro-key-concepts
15. Leveraging Snowflake for Real-Time Business Intelligence and Analytics - ResearchGate, accessed June 3, 2025, https://www.researchgate.net/publication/389434895_Leveraging_Snowflake_for_Real-Time_Business_Intelligence_and_Analytics
16. Building a Data Lake Architecture with Apache Hive: A Comprehensive Guide, accessed June 3, 2025, https://www.sparkcodehub.com/hive/misc/data-lake-architecture
17. Practical use cases of generative AI in AML compliance - Salv, accessed June 3, 2025, https://salv.com/blog/generative-AI-use-cases-AML-compliance/
18. BSA/AML Manual - Appendices - Appendix F – Money Laundering and Terrorist Financing Red Flags, accessed June 3, 2025, https://bsaaml.ffiec.gov/manual/Appendices/07
19. Alerts/Advisories/Notices/Bulletins/Fact Sheets | FinCEN.gov, accessed June 3, 2025, https://www.fincen.gov/resources/advisoriesbulletinsfact-sheets

20. AML Compliance for RIAs: FinCEN Rule 2026 Transaction Monitoring Guide - Flagright, accessed June 3, 2025, https://www.flagright.com/post/transaction-monitoring-for-rias-fincen-aml-rule
21. "Black and grey" lists - FATF, accessed June 3, 2025, https://www.fatf-gafi.org/en/countries/black-and-grey-lists.html
22. The FATF Recommendations, accessed June 3, 2025, https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Fatf-recommendations.html
23. 10 Principles for Better Transaction Monitoring | Analytics Magazine - PubsOnLine, accessed June 3, 2025, https://pubsonline.informs.org/do/10.1287/LYTX.2025.01.14/full/
24. Understanding the Wolfsberg AML Principles: An Overview and Impact on Global Financial Compliance | sanctions.io, accessed June 3, 2025, https://www.sanctions.io/blog/wolfsberg-aml-principles
25. The 10 Most Common AML Red Flags to Watch Out for in 2025 - Sumsub, accessed June 3, 2025, https://sumsub.com/blog/the-10-most-common-aml-red-flags-complete-guide/
26. Money Laundering Transactions Chronology Analysis using Artificial Intelligence - RIT Digital Institutional Repository, accessed June 3, 2025, https://repository.rit.edu/cgi/viewcontent.cgi?article=13141&context=theses
27. AML Prediction - Kaggle, accessed June 3, 2025, https://www.kaggle.com/code/gbiamgaurav/aml-prediction
28. Synthetic Data Sets | IBM, accessed June 3, 2025, https://www.ibm.com/products/synthetic-data-sets
29. Synthetic Data for Anti-Money Laundering | The Alan Turing Institute, accessed June 3, 2025, https://www.turing.ac.uk/research/research-projects/synthetic-data-anti-money-laundering
30. Synthetic Data in AML: Crafting Realistic Scenarios for Enhanced Compliance Testing, accessed June 3, 2025, https://lucinity.com/blog/synthetic-data-in-aml-crafting-realistic-scenarios-for-enhanced-compliance-testing
31. eprints.bournemouth.ac.uk, accessed June 3, 2025, https://eprints.bournemouth.ac.uk/40982/1/Full_IEEE_Dataset_Conference_Paper%20%284%29.pdf
32. Synthetic Financial Data Generation for Fraud Detection - GitHub, accessed June 3, 2025, https://github.com/sonu-gupta/Synthetic-financial-data
33. Enhancing Anti-Money Laundering: Development of a Synthetic Transaction Monitoring Dataset. - BURO, accessed June 3, 2025, https://eprints.bournemouth.ac.uk/40982/
34. Statistical in Sights in to Anti-Money Laundering: Analyzing Large-Scale Financial Transactions - International Journal of Engineering Research & Technology, accessed June 3, 2025, https://www.ijert.org/statistical-in-sights-in-to-anti-money-laundering-analyzing-large-scale-financial-transactions

35. zhihuat/Collaborative-AML - GitHub, accessed June 3, 2025, https://github.com/zhihuat/Collaborative-AML
36. IBM Transactions for Anti Money Laundering (AML) - Kaggle, accessed June 3, 2025, https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml/data
37. paysim dataset - Kaggle, accessed June 3, 2025, https://www.kaggle.com/datasets/mtalaltariq/paysim-data/versions/1
38. Fraud Detection Paysim data - Kaggle, accessed June 3, 2025, https://www.kaggle.com/code/dhanyabahadur/fraud-detection-paysim-data
39. Paysim data set with network features - Kaggle, accessed June 3, 2025, https://www.kaggle.com/datasets/harinip/paysim-data-set-with-network-features
40. Synthetic Financial Datasets For Fraud Detection - Kaggle, accessed June 3, 2025, https://www.kaggle.com/datasets/ealaxi/paysim1
41. accessed December 31, 1969, https://www.kaggle.com/datasets/ealaxi/paysim1/discussion/136007
42. How You Can Migrate AWS Glue to AWS Snowflake? - TechAhead, accessed June 3, 2025, https://www.techaheadcorp.com/blog/how-you-can-migrate-aws-glue-to-aws-snowflake/
43. Using the Spark Connector | Snowflake Documentation, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/spark-connector-use
44. Snowflake connections - AWS Glue, accessed June 3, 2025, https://docs.aws.amazon.com/glue/latest/dg/aws-glue-programming-etl-connect-snowflake-home.html
45. How we migrated from AWS Glue to Snowflake dbt - Theodo Data & AI, accessed June 3, 2025, https://data-ai.theodo.com/en/technical-blog/how-we-migrated-from-aws-glue-to-snowflake-and-dbt
46. Apache Spark on EMR: Setup & Optimization Guide (2025) - Chaos Genius, accessed June 3, 2025, https://www.chaosgenius.io/blog/apache-spark-on-emr/
47. Amazon EMR: A Complete Hands-On Guide for Beginners - DataCamp, accessed June 3, 2025, https://www.datacamp.com/tutorial/amazon-emr
48. AWS EMR Architecture 101—Core Features & Components (2025) - Chaos Genius, accessed June 3, 2025, https://www.chaosgenius.io/blog/aws-emr-architecture/
49. Anti-money laundering in Python - Deepnote, accessed June 3, 2025, https://deepnote.com/guides/tutorials/anti-money-laundering-in-python
50. Big Data-Driven Distributed Machine Learning for Scalable Credit Card Fraud Detection Using PySpark, XGBoost, and CatBoost - MDPI, accessed June 3, 2025, https://www.mdpi.com/2079-9292/14/9/1754
51. (PDF) Feature Engineering for Transaction Anomalies - ResearchGate, accessed June 3, 2025, https://www.researchgate.net/publication/388026453_Feature_Engineering_for_Tr

ansaction_Anomalies
52. CREATE STORAGE INTEGRATION - Snowflake Documentation, accessed June 3, 2025, https://docs.snowflake.com/en/sql-reference/sql/create-storage-integration
53. Configuring Storage Integrations Between Snowflake and AWS S3 - InterWorks, accessed June 3, 2025, https://interworks.com/blog/2023/02/14/configuring-storage-integrations-between-snowflake-and-aws-s3/
54. Table Design Considerations | Snowflake Documentation, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/table-considerations
55. Data Warehouse Design | Snowflake Guides, accessed June 3, 2025, https://www.snowflake.com/guides/data-warehouse-design/
56. How Strategic AML Datasets Drive Compliance | The Perfect Merchant, accessed June 3, 2025, https://theperfectmerchant.com/how-strategic-aml-datasets-drive-compliance/
57. Working with Materialized Views - Snowflake Documentation, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/views-materialized
58. Fraud Detection & Financial Crimes - Snowflake, accessed June 3, 2025, https://www.snowflake.com/en/solutions/industries/financial-services/fraud-detection-and-financial-crimes/
59. Using AI Within Snowflake For Everyday Analytics - YouTube, accessed June 3, 2025, https://www.youtube.com/watch?v=en_HbcAKvp4
60. Anomaly Detection in Python with Isolation Forest - DigitalOcean, accessed June 3, 2025, https://www.digitalocean.com/community/tutorials/anomaly-detection-isolation-forest
61. Isolation Forest Guide: Explanation and Python Implementation - DataCamp, accessed June 3, 2025, https://www.datacamp.com/tutorial/isolation-forest
62. Clustering: KMeans in PySpark: A Comprehensive Guide - SparkCodeHub, accessed June 3, 2025, https://www.sparkcodehub.com/pyspark/mllib/kmeans
63. Outliers Detection in PySpark #3 – K-means | Awaiting Bits, accessed June 3, 2025, https://blog.zhaytam.com/2019/08/06/outliers-detection-in-pyspark-3-k-means/
64. Clustering - Spark 4.0.0 Documentation, accessed June 3, 2025, https://spark.apache.org/docs/latest/ml-clustering.html
65. Anomaly detection using embeddings and GenAI - Databricks Community, accessed June 3, 2025, https://community.databricks.com/t5/technical-blog/anomaly-detection-using-embeddings-and-genai/ba-p/95564
66. Anomaly detection with PCA in Spark - Stack Overflow, accessed June 3, 2025, https://stackoverflow.com/questions/49530351/anomaly-detection-with-pca-in-spark
67. Build a Generative BI Dashboard with Amazon QuickSight and Amazon Q - Snowflake Quickstarts, accessed June 3, 2025, https://quickstarts.snowflake.com/guide/quickstart-generative-bi-quicksight/index.html?index=..%2F..index

68. How to connect Amazon QuickSight to Snowflake Data - CData Software, accessed June 3, 2025, https://www.cdata.com/kb/tech/snowflake-cloud-quicksight.rst
69. Creating Alerts - Amazon QuickSight - AWS Documentation, accessed June 3, 2025, https://docs.aws.amazon.com/quicksight/latest/user/threshold-alerts-creating.html
70. Configuring an QuickSight dashboard for the Amazon Chime SDK, accessed June 3, 2025, https://docs.aws.amazon.com/chime-sdk/latest/dg/quicksight-setup-setup.html
71. Managing Threshold Alerts - Amazon QuickSight - AWS Documentation, accessed June 3, 2025, https://docs.aws.amazon.com/quicksight/latest/user/threshold-alerts-managing.html
72. Threshold Alerts - YouTube, accessed June 3, 2025, https://www.youtube.com/watch?v=t58lSlu3xs8
73. Setting up alerts based on data in Snowflake, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/alerts
74. Configure IAM service roles for Amazon EMR permissions to AWS services and resources, accessed June 3, 2025, https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-iam-roles.html
75. Configure IAM roles for EMRFS requests to Amazon S3 - AWS Documentation, accessed June 3, 2025, https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-emrfs-iam-roles.html
76. Read and write Apache Iceberg tables using AWS Lake Formation hybrid access mode, accessed June 3, 2025, https://aws.amazon.com/blogs/big-data/read-and-write-apache-iceberg-tables-using-aws-lake-formation-hybrid-access-mode/
77. Accessing Amazon S3 tables with Amazon EMR - Amazon Simple Storage Service, accessed June 3, 2025, https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-tables-integrating-emr.html
78. AWS IAM policies - PuppyGraph Docs, accessed June 3, 2025, https://docs.puppygraph.com/reference/cloud_integrations/aws/aws-iam-policies/
79. Write a Simple IAM Policy, accessed June 3, 2025, https://slaw.securosis.com/p/write-simple-iam-policy
80. Configure a catalog integration for AWS Glue - Snowflake Documentation, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/tables-iceberg-configure-catalog-integration-glue
81. Identity-based policy examples for AWS Glue, accessed June 3, 2025, https://docs.aws.amazon.com/glue/latest/dg/security_iam_id-based-policy-examples.html

82. Setting up IAM permissions for AWS Glue, accessed June 3, 2025, https://docs.aws.amazon.com/glue/latest/dg/set-up-iam.html
83. Option 2: Configuring an AWS IAM role to access Amazon S3 — Deprecated, accessed June 3, 2025, https://docs.snowflake.com/en/user-guide/data-load-s3-config-aws-iam-role
84. AML Data Sources - SEON Docs, accessed June 3, 2025, https://docs.seon.io/knowledge-base/data-enrichment/aml-data-sources