AWS Data Engineer - Associate Exam

Past Exam Questions with Explanations

By: Hassan Ashas

Systems Limited

Question # 1

Date: February 06, 2024

A data engineer is configuring an AWS Glue job to read data from an Amazon S3 bucket. The data engineer has set up the necessary AWS Glue connection details and an associated IAM role. However, when the data engineer attempts to run the AWS Glue job, the data engineer receives an error message that indicates that there are problems with the Amazon S3 VPC gateway endpoint.

The data engineer must resolve the error and connect the AWS Glue job to the S3 bucket.

Which solution will meet this requirement?

A. Update the AWS Glue security group to allow inbound traffic from the Amazon S3 VPC gateway endpoint.
B. Configure an S3 bucket policy to explicitly grant the AWS Glue job permissions to access the S3 bucket.
C. Review the AWS Glue job code to ensure that the AWS Glue connection details include a fully qualified domain name.
D. Verify that the VPC's route table includes inbound and outbound routes for the Amazon S3 VPC gateway endpoint.

Correct Answer:

A. Update the AWS Glue security group to allow inbound traffic from the Amazon S3 VPC gateway endpoint.

B. Configure an S3 bucket policy to explicitly grant the AWS Glue job permissions to access the S3 bucket.

C. Review the AWS Glue job code to ensure that the AWS Glue connection details include a fully qualified domain name.

D. Verify that the VPC's route table includes inbound and outbound routes for the Amazon S3 VPC gateway endpoint.

Explanation:

In the scenario described, the data engineer is trying to run an AWS Glue job to access data stored in an Amazon S3 bucket. The error message received indicates issues related to the Amazon S3 VPC gateway endpoint. This suggests that the problem lies in the network configuration rather than permissions or the Glue job code itself.

The correct answer, D, involves verifying the VPC's route table to ensure that it includes the appropriate routes for the Amazon S3 VPC gateway endpoint. Here's why this is the correct solution:

1. VPC Gateway Endpoint Basics: A VPC gateway endpoint allows resources in a VPC to communicate with supported AWS services, such as Amazon S3, without needing to traverse the public internet. This is particularly important for security and performance reasons.

2. Route Table Configuration: For a VPC gateway endpoint to function correctly, the VPC's route table must be properly configured. This means that the route table should have entries directing traffic destined for Amazon S3 through the gateway endpoint. Without these routes, traffic from the AWS Glue job will not reach the S3 service, resulting in connectivity issues.

3. Common Misconfiguration: A frequent error when configuring VPC endpoints is neglecting to update the route tables appropriately. This can prevent the necessary network traffic from flowing through the endpoint, leading to the kind of error experienced by the data engineer.

4. Corrective Action: By verifying and updating the route tables to include the correct routes for the S3 VPC gateway endpoint, the network path is established. This allows the AWS Glue job to access the S3 bucket as intended.

The other options do not address the core issue indicated by the error message:

- Option A involves security group settings, which are not relevant for gateway endpoints, as these endpoints do not require inbound traffic rules. - Option B focuses on bucket policies, which govern permissions, not network connectivity. - Option C pertains to the job code configuration, which is unrelated to the network path issue described.

In summary, option D is the correct solution because it directly addresses the network configuration issue by ensuring the VPC route tables are set up to facilitate traffic to the S3 service via the VPC gateway endpoint.

Question # 2

Date:

A retail company has a customer data hub in an Amazon S3 bucket. Employees from many countries use the data hub to support company-wide analytics. A governance team must ensure that the company's data analysts can access data only for customers who are within the same country as the analysts.

Which solution will meet these requirements with the LEAST operational effort?

A. Create a separate table for each country's customer data. Provide access to each analyst based on the country that the analyst serves.
B. Register the S3 bucket as a data lake location in AWS Lake Formation. Use the Lake Formation row-level security features to enforce the company's access policies.
C. Move the data to AWS Regions that are close to the countries where the customers are. Provide access to each analyst based on the country that the analyst serves.
D. Load the data into Amazon Redshift. Create a view for each country. Create separate IAM roles for each country to provide access to data from each country. Assign the appropriate roles to the analysts.

Correct Answer:

A. Create a separate table for each country's customer data. Provide access to each analyst based on the country that the analyst serves.

B. Register the S3 bucket as a data lake location in AWS Lake Formation. Use the Lake Formation row-level security features to enforce the company's access policies.

C. Move the data to AWS Regions that are close to the countries where the customers are. Provide access to each analyst based on the country that the analyst serves.

D. Load the data into Amazon Redshift. Create a view for each country. Create separate IAM roles for each country to provide access to data from each country. Assign the appropriate roles to the analysts.

Explanation:

The correct answer is B: Register the S3 bucket as a data lake location in AWS Lake Formation. Use the Lake Formation row-level security features to enforce the company's access policies.

Here's why this solution is the best choice with the least operational effort:

1. AWS Lake Formation Overview: AWS Lake Formation is a service designed to help set up a secure data lake in days. It simplifies the process of collecting, cataloging, and securing data in a data lake. One of its key features is the ability to manage access control at a very granular level, including row-level security.

2. Row-Level Security: In this scenario, the governance team needs to ensure that data analysts only access customer data relevant to their respective countries. Lake Formation's row-level security allows you to define permissions that restrict access to specific rows within a dataset based on certain criteria, such as the country of the customer. This means you can enforce access policies based directly on the data itself without needing to create separate datasets or complex access controls.

3. Operational Simplicity: By using Lake Formation, you centralize the data in a single Amazon S3 bucket, which reduces the complexity of managing multiple datasets or data locations. The row-level security is implemented through policies in Lake Formation, which are easier to manage and update than having separate datasets or deploying different infrastructure components for each country's data.

4. Scalability and Maintainability: Lake Formation's approach is scalable and maintainable. As new data comes in or as analysts' assignments change, the governance policies can be updated without needing to restructure the data or move it to different locations. This reduces the operational overhead as the company grows or as requirements evolve.

5. Compared to Other Options: - Option A involves creating separate tables for each country, which increases the complexity of data management and can lead to data duplication. - Option C suggests moving data to different AWS Regions, which is inefficient and could lead to increased costs and complexity in data management. - Option D requires setting up a Redshift cluster and managing multiple views and IAM roles, which involves more infrastructure and is more complex to maintain.

In summary, using AWS Lake Formation with row-level security provides a streamlined, efficient, and secure way to meet the governance requirements with minimal operational effort compared to other options.

Question # 3

Date:

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform.

The company wants to minimize the effort and time required to incorporate third-party datasets.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.
B. Use API calls to access and integrate third-party datasets from AWS DataSync.
C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.
D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

Correct Answer:

A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.

B. Use API calls to access and integrate third-party datasets from AWS DataSync.

C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.

D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

Explanation:

The correct answer is A: Use API calls to access and integrate third-party datasets from AWS Data Exchange.

Here's why this is the best solution with the least operational overhead:

1. AWS Data Exchange: This service is specifically designed to make it easy to find, subscribe to, and use third-party data in the AWS ecosystem. By using AWS Data Exchange, the media company can directly subscribe to datasets offered by third-party data providers. This service simplifies the acquisition and integration process as it provides a marketplace where data is readily available and can be accessed through standardized APIs.

2. Ease of Integration: With AWS Data Exchange, once the company subscribes to the desired datasets, it can easily integrate this data into its existing analytics platform using API calls. This minimizes the need for complex data pipelines or custom integrations, reducing both time and effort.

3. Operational Overhead: Since AWS Data Exchange manages data subscriptions, updates, and delivery, the operational overhead for the media company is significantly reduced. The company doesn't need to manage data transfer processes or worry about data format compatibility.

4. Data Security and Compliance: AWS Data Exchange ensures that data is delivered securely, and it handles compliance and billing, which further reduces the operational responsibilities of the company.

5. Alternatives: - AWS DataSync (Option B) is primarily used for data transfer between on-premises storage and AWS, not for accessing third-party datasets. - Amazon Kinesis Data Streams (Options C and D) is used for real-time data streaming, not for accessing third-party datasets. Additionally, AWS CodeCommit and Amazon ECR are not typical sources for third-party datasets meant for analytics.

In summary, AWS Data Exchange is purpose-built for accessing and using third-party datasets with minimal operational complexity, making it the optimal choice for the media company's needs.

Question # 4

Date: January 18, 2024

A financial company wants to implement a data mesh. The data mesh must support centralized data governance, data analysis, and data access control. The company has decided to use AWS Glue for data catalogs and extract, transform, and load (ETL) operations.

Which combination of AWS services will implement a data mesh? (Choose two.)

A. Use Amazon Aurora for data storage. Use an Amazon Redshift provisioned cluster for data analysis.
B. Use Amazon S3 for data storage. Use Amazon Athena for data analysis.
C. Use AWS Glue DataBrew for centralized data governance and access control.
D. Use Amazon RDS for data storage. Use Amazon EMR for data analysis.
E. Use AWS Lake Formation for centralized data governance and access control.

Correct Answer:

A. Use Amazon Aurora for data storage. Use an Amazon Redshift provisioned cluster for data analysis.

B. Use Amazon S3 for data storage. Use Amazon Athena for data analysis.

C. Use AWS Glue DataBrew for centralized data governance and access control.

D. Use Amazon RDS for data storage. Use Amazon EMR for data analysis.

E. Use AWS Lake Formation for centralized data governance and access control.

Explanation:

Certainly! To understand why options B and E are the correct choices for implementing a data mesh with centralized data governance, data analysis, and data access control using AWS services, let's break down the solution:

B. Use Amazon S3 for data storage. Use Amazon Athena for data analysis.

- Amazon S3 (Simple Storage Service): It is a highly scalable and durable object storage service. In the context of a data mesh, S3 acts as a foundational layer for storing a vast amount of data in a cost-effective and scalable manner. It provides the flexibility to store different types of data from various domains in a unified data lake, which is essential for a data mesh architecture.

- Amazon Athena: This is an interactive query service that allows you to analyze data directly in Amazon S3 using standard SQL. Athena is serverless, meaning no infrastructure needs to be managed, and it can scale automatically, making it a great choice for data analysis in a data mesh. It allows users across different domains to perform ad-hoc analysis without the need for complex ETL pipelines.

E. Use AWS Lake Formation for centralized data governance and access control.

- AWS Lake Formation: This service simplifies the process of setting up a secure data lake in days. It provides centralized data governance features, including fine-grained access control and audit capabilities. Lake Formation integrates seamlessly with AWS Glue, making it an ideal choice for managing permissions and access control across a data mesh. It ensures that data from various domains can be securely shared and governed according to company policies.

Why B and E are Correct:

- Centralized Data Governance: AWS Lake Formation offers robust features for defining and enforcing security policies, auditing usage, and managing access to data stored in S3. This aligns perfectly with the need for centralized data governance in a data mesh.

- Data Storage and Analysis: Using Amazon S3 and Athena provides a flexible, scalable, and cost-effective solution for storing and analyzing data. They support the decentralized nature of a data mesh by allowing different teams to store and process data independently while still adhering to centralized governance policies.

Together, these services (S3, Athena, and Lake Formation) provide the essential components needed to implement a data mesh architecture on AWS, focusing on centralized governance, access control, and efficient data analysis.

Question # 5

Date:

A data engineer maintains custom Python scripts that perform a data formatting process that many AWS Lambda functions use. When the data engineer needs to modify the Python scripts, the data engineer must manually update all the Lambda functions.

The data engineer requires a less manual way to update the Lambda functions.

Which solution will meet this requirement?

A. Store a pointer to the custom Python scripts in the execution context object in a shared Amazon S3 bucket.
B. Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions.
C. Store a pointer to the custom Python scripts in environment variables in a shared Amazon S3 bucket.
D. Assign the same alias to each Lambda function. Call reach Lambda function by specifying the function's alias.

Correct Answer:

A. Store a pointer to the custom Python scripts in the execution context object in a shared Amazon S3 bucket.

B. Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions.

C. Store a pointer to the custom Python scripts in environment variables in a shared Amazon S3 bucket.

D. Assign the same alias to each Lambda function. Call reach Lambda function by specifying the function's alias.

Explanation:

The problem here is that the data engineer is manually updating multiple AWS Lambda functions whenever there's a change in the custom Python scripts used by these functions. This is time-consuming and prone to errors. The goal is to find a solution that allows the engineer to update the scripts in a more centralized and efficient manner.

The correct answer is B: "Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions."

Here's why this solution is ideal:

1. Lambda Layers Overview: AWS Lambda layers allow you to package additional content, such as libraries, dependencies, or custom code, separately from your main function code. These layers can be shared across multiple Lambda functions. This means that when you update the layer, all functions using that layer can access the new version without needing individual updates.

2. Centralized Management: By packaging the custom Python scripts into a Lambda layer, you centralize the management of these scripts. When a script update is needed, you only need to update the layer itself, rather than each Lambda function. This significantly reduces manual work and the possibility of errors.

3. Version Control: Lambda layers support versioning. When you update a layer, you create a new version. Your Lambda functions can then be configured to use the latest version or a specific version of the layer. This allows you to test new versions safely before fully deploying them.

4. Reusability and Consistency: Using layers ensures that the same version of the script is used across all functions, maintaining consistency. It also encourages reusability of code, which is a best practice in software development.

5. Separation of Concerns: By separating the script logic from the core function logic, you adhere to the principle of separation of concerns. This makes your Lambda function code cleaner and easier to maintain.

In contrast, the other options do not provide the same level of efficiency or ease of management: - Option A suggests using S3 and the execution context object, which could complicate the execution and doesn't directly address the need for centralized updates. - Option C is similar to A, with environment variables pointing to S3, which again doesn't streamline the update process. - Option D involves using aliases for Lambda functions, which doesn't solve the problem of centralized script updates at all.

Therefore, packaging the scripts into Lambda layers (Option B) is the most effective solution for reducing manual updates and maintaining consistency across multiple Lambda functions.

Question # 6

Date: January 18, 2024

A company created an extract, transform, and load (ETL) data pipeline in AWS Glue. A data engineer must crawl a table that is in Microsoft SQL Server. The data engineer needs to extract, transform, and load the output of the crawl to an Amazon S3 bucket. The data engineer also must orchestrate the data pipeline.

Which AWS service or feature will meet these requirements MOST cost-effectively?

A. AWS Step Functions
B. AWS Glue workflows
C. AWS Glue Studio
D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Correct Answer:

A. AWS Step Functions

B. AWS Glue workflows

C. AWS Glue Studio

D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Explanation:

The correct answer to the question is B: AWS Glue workflows. Let's break down why this is the most cost-effective choice for the scenario described.

1. AWS Glue Overview: - AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy to prepare and load data for analytics. It includes a data catalog, a scheduler, an ETL engine, and workflows to manage the ETL process.

2. Requirements: - The data engineer needs to extract data from a Microsoft SQL Server, transform it, and load it into an Amazon S3 bucket. Additionally, they need to orchestrate this data pipeline.

3. AWS Glue Workflows: - AWS Glue workflows are specifically designed to manage and automate the execution of ETL jobs. They allow you to define a series of interconnected jobs, crawlers, and triggers as a single workflow. This orchestration capability means that you can automate the entire process of extracting, transforming, and loading data without needing to manage the underlying infrastructure. - Glue workflows integrate directly with other Glue components like crawlers and ETL jobs, making it easy to set up a complete data pipeline. - Since AWS Glue is serverless, you only pay for the resources you consume, which can be more cost-effective than other orchestration tools if your workloads are variable or infrequent.

4. Comparison with Other Options: - AWS Step Functions (Option A): While Step Functions is a powerful orchestration service that can coordinate various AWS services, it is more general-purpose and might require more custom setup to integrate AWS Glue jobs efficiently. It could be more costly due to the additional configuration and invocation charges. - AWS Glue Studio (Option C): Glue Studio is a visual interface for creating, running, and monitoring ETL jobs. It's excellent for designing ETL jobs but doesn't inherently provide orchestration capabilities like Glue workflows. - Amazon MWAA (Option D): Managed Workflows for Apache Airflow is a managed service for orchestrating workflows using Apache Airflow. While it's a robust tool for complex workflows, it may be overkill for simple ETL pipelines and could incur higher costs due to its broader capabilities and resource requirements.

In summary, AWS Glue workflows are best suited for this task as they are specifically designed to orchestrate and automate ETL processes within AWS Glue. This makes them a cost-effective and efficient choice for managing the ETL pipeline described in the question.

Question # 7

Date: January 18, 2024

A financial services company stores financial data in Amazon Redshift. A data engineer wants to run real-time queries on the financial data to support a web-based trading application. The data engineer wants to run the queries from within the trading application.

Which solution will meet these requirements with the LEAST operational overhead?

A. Establish WebSocket connections to Amazon Redshift.
B. Use the Amazon Redshift Data API.
C. Set up Java Database Connectivity (JDBC) connections to Amazon Redshift.
D. Store frequently accessed data in Amazon S3. Use Amazon S3 Select to run the queries.

Correct Answer:

A. Establish WebSocket connections to Amazon Redshift.

B. Use the Amazon Redshift Data API.

C. Set up Java Database Connectivity (JDBC) connections to Amazon Redshift.

D. Store frequently accessed data in Amazon S3. Use Amazon S3 Select to run the queries.

Explanation:

The correct answer is B: "Use the Amazon Redshift Data API."

Here's why this option is the most suitable for meeting the requirements with the least operational overhead:

1. Purpose of the Redshift Data API: The Amazon Redshift Data API is designed to simplify access to Amazon Redshift. It allows you to run SQL queries directly from your applications without needing to manage database connections. This is particularly useful for applications that need to perform real-time queries, like a web-based trading application.

2. Real-Time Query Execution: By using the Redshift Data API, the data engineer can execute queries on the financial data stored in Amazon Redshift directly from the trading application. It supports both synchronous and asynchronous query execution, making it well-suited for real-time operations where latency might be a concern.

3. Reduced Operational Overhead: The Data API abstracts much of the complexity associated with managing connections to a database. Traditional methods like JDBC connections require managing and maintaining database connection pools, which can add operational overhead. The Data API, on the other hand, handles these connections for you, reducing the need for additional infrastructure or complex connection management.

4. Security and Integration: The Redshift Data API integrates with AWS Identity and Access Management (IAM), allowing secure access to Redshift data without embedding database credentials in your application code. This enhances security and simplifies the integration process.

5. Scalability and Management: The API is serverless, meaning AWS manages the scalability and availability of the service, which reduces the need for the data engineer to manage infrastructure. This is especially advantageous for applications that may experience variable loads, such as a trading platform that might see spikes in activity.

In contrast, the other options either require more complex setup and maintenance (like establishing WebSocket or JDBC connections) or do not directly support real-time query execution on Redshift data (like using Amazon S3 Select, which is more suited for querying data stored in S3 rather than Redshift).

Overall, using the Amazon Redshift Data API provides a streamlined, efficient way to integrate real-time queries into an application with minimal operational overhead, making it the best choice for this scenario.

Question # 8

Date: January 18, 2024

A company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account.

Which solution will meet these requirements?

A. Create an S3 bucket for each use case. Create an S3 bucket policy that grants permissions to appropriate individual IAM users. Apply the S3 bucket policy to the S3 bucket.
B. Create an Athena workgroup for each use case. Apply tags to the workgroup. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.
C. Create an IAM role for each use case. Assign appropriate permissions to the role for each use case. Associate the role with Athena.
D. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use case. Apply the resource policy to the specific tables that Athena uses.

Correct Answer:

A. Create an S3 bucket for each use case. Create an S3 bucket policy that grants permissions to appropriate individual IAM users. Apply the S3 bucket policy to the S3 bucket.

B. Create an Athena workgroup for each use case. Apply tags to the workgroup. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.

C. Create an IAM role for each use case. Assign appropriate permissions to the role for each use case. Associate the role with Athena.

D. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use case. Apply the resource policy to the specific tables that Athena uses.

Explanation:

The correct answer is B: Create an Athena workgroup for each use case. Apply tags to the workgroup. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.

Here's a detailed explanation of why this is the best solution:

1. Athena Workgroups: Amazon Athena workgroups are a feature designed to separate users, teams, or applications within the same AWS account based on their specific use cases. By creating different workgroups, you can segregate query workloads and manage permissions for each workgroup independently. This is crucial when you need to control who can run queries and access query histories, as well as manage costs and resources among different teams or applications.

2. Tagging and IAM Policies: By applying tags to each workgroup, you can easily manage and organize them according to their specific use cases. AWS Identity and Access Management (IAM) policies can be created to use these tags for fine-grained access control. This means you can specify which users or groups have permission to access specific workgroups based on the tags assigned to them, effectively controlling access to query processes and histories.

3. Separation of Duties: Using workgroups aligns well with the requirement to separate query processes and access to query history among users, teams, and applications. Each workgroup can have its own configuration, such as query result location, encryption settings, and more, ensuring complete isolation of data and query activities.

4. Scalability and Flexibility: Creating workgroups allows the company to scale its operations easily. As new use cases arise or existing ones evolve, new workgroups can be created or existing ones modified without impacting other teams or applications. This flexibility is essential in a dynamic environment where data needs and team structures may change frequently.

5. Cost Management: Workgroups in Athena can also help manage costs by tracking the usage and costs associated with each workgroup. You can monitor queries run by each workgroup and use AWS Cost Explorer to analyze and allocate costs appropriately.

In contrast, the other options do not address the requirements as effectively:

- Option A focuses on S3 bucket policies, which do not directly manage Athena query permissions or histories. - Option C involves IAM roles, but these do not provide the same level of query process separation and history access control as workgroups. - Option D suggests using AWS Glue Data Catalog resource policies, which are more suitable for controlling access to metadata rather than managing query processes and histories in Athena.

Thus, option B provides a comprehensive solution for managing permissions and separating query processes in Amazon Athena.

Question # 9

Date: February 01, 2024

A data engineer needs to schedule a workflow that runs a set of AWS Glue jobs every day. The data engineer does not require the Glue jobs to run or finish at a specific time.

Which solution will run the Glue jobs in the MOST cost-effective way?

A. Choose the FLEX execution class in the Glue job properties.
B. Use the Spot Instance type in Glue job properties.
C. Choose the STANDARD execution class in the Glue job properties.
D. Choose the latest version in the GlueVersion field in the Glue job properties.

Correct Answer:

A. Choose the FLEX execution class in the Glue job properties.

B. Use the Spot Instance type in Glue job properties.

C. Choose the STANDARD execution class in the Glue job properties.

D. Choose the latest version in the GlueVersion field in the Glue job properties.

Explanation:

Certainly! Let's break down why choosing the FLEX execution class (Option A) is the most cost-effective solution for scheduling AWS Glue jobs that do not require running or finishing at a specific time.

AWS Glue is a serverless data integration service that makes it easier to discover, prepare, and combine data for analytics, machine learning, and application development. When it comes to running Glue jobs, cost-effectiveness is a key consideration, especially if the jobs can be run flexibly without time constraints.

1. FLEX Execution Class: The FLEX execution class is designed to be more cost-effective than the STANDARD execution class. It achieves this by allowing AWS Glue to utilize spare capacity in the AWS infrastructure. This means that jobs may start with a delay compared to the STANDARD execution class, but they will be cheaper because they take advantage of these available resources.

2. Cost-Effectiveness: Since the data engineer does not require the Glue jobs to start or finish at a specific time, the potential delay associated with the FLEX execution class is not a concern. This flexibility enables AWS to pass on cost savings to the customer.

3. AWS Glue Pricing: The pricing model for AWS Glue is based on the amount of data processed and the time taken by the jobs. By choosing the FLEX execution class, the data engineer can significantly reduce costs without impacting the operation of the jobs, given that timing is not a constraint.

4. Comparison with Other Options: - Spot Instances (Option B): AWS Glue does not use Spot Instances in the traditional EC2 sense. Spot Instances refer to a purchasing option for EC2 where users can bid on spare compute capacity at reduced rates, but this is not directly applicable to AWS Glue jobs. - STANDARD Execution Class (Option C): The STANDARD execution class offers predictable start times but at a higher cost compared to FLEX. It is suitable when jobs need to start immediately or at a specific time. - Latest Version in GlueVersion (Option D): Choosing the latest version of AWS Glue might offer performance improvements or new features, but it doesn't inherently contribute to cost savings. In fact, newer versions might have higher costs associated with them if they include more advanced capabilities.

In summary, the FLEX execution class offers the best cost-efficiency for running AWS Glue jobs when there is no requirement for the jobs to run or complete at specific times. This makes Option A the correct choice for the given scenario.

Question # 10

Date: January 18, 2024

A data engineer needs to create an AWS Lambda function that converts the format of data from .csv to Apache Parquet. The Lambda function must run only if a user uploads a .csv file to an Amazon S3 bucket.

Which solution will meet these requirements with the LEAST operational overhead?

A. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
B. Create an S3 event notification that has an event type of s3:ObjectTagging:* for objects that have a tag set to .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
C. Create an S3 event notification that has an event type of s3:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
D. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set an Amazon Simple Notification Service (Amazon SNS) topic as the destination for the event notification. Subscribe the Lambda function to the

SNS topic.

Correct Answer:

A. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.

B. Create an S3 event notification that has an event type of s3:ObjectTagging:* for objects that have a tag set to .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.

C. Create an S3 event notification that has an event type of s3:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.

D. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set an Amazon Simple Notification Service (Amazon SNS) topic as the destination for the event notification. Subscribe the Lambda function to the SNS topic.

Explanation:

The correct answer is A.

Here's why this solution is ideal for the scenario described, focusing on achieving the goal with the least operational overhead:

1. S3 Event Notification: By configuring an S3 event notification with the event type `s3:ObjectCreated:`, you ensure that the notification is triggered any time a new object is created in the S3 bucket. This is essential because the Lambda function should run whenever a new .csv file is uploaded.

2. Filter Rule: The use of a filter rule to check that the suffix of the file is `.csv` ensures that only CSV files trigger the Lambda function. This is a crucial optimization because it prevents unnecessary execution of the Lambda function for non-CSV files, thus saving on processing time and costs.

3. Direct Invocation of Lambda Function: Setting the Amazon Resource Name (ARN) of the Lambda function as the direct destination for the event notification minimizes operational overhead. This approach eliminates the need for additional services in the data processing workflow, such as Amazon SNS or SQS, which would add complexity and potential latency to the process.

4. Simplicity and Efficiency: This solution is both simple and efficient. It leverages S3's built-in capability to notify Lambda directly, avoiding unnecessary infrastructure management. This direct approach ensures the fastest and most straightforward execution path from file upload to processing.

5. Cost-Effectiveness: By using a direct trigger from S3 to Lambda, the architecture minimizes costs. There's no need for additional services that could incur separate charges, and the Lambda function only runs when necessary, avoiding unnecessary compute charges.

In summary, option A provides a streamlined, efficient, and cost-effective solution that directly meets the requirement of executing a Lambda function upon the upload of a .csv file to an S3 bucket, all while maintaining minimal operational overhead.

Question # 11

Date: February 02, 2024

A data engineer must orchestrate a data pipeline that consists of one AWS Lambda function and one AWS Glue job. The solution must integrate with AWS services.

Which solution will meet these requirements with the LEAST management overhead?

A. Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job.
B. Use an Apache Airflow workflow that is deployed on an Amazon EC2 instance. Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.
C. Use an AWS Glue workflow to run the Lambda function and then the AWS Glue job.
D. Use an Apache Airflow workflow that is deployed on Amazon Elastic Kubernetes Service (Amazon EKS). Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

Correct Answer:

A. Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job.

B. Use an Apache Airflow workflow that is deployed on an Amazon EC2 instance. Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

C. Use an AWS Glue workflow to run the Lambda function and then the AWS Glue job.

D. Use an Apache Airflow workflow that is deployed on Amazon Elastic Kubernetes Service (Amazon EKS). Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

Explanation:

The correct answer is A: Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job.

Here's why this solution is the best choice with the least management overhead:

1. AWS Step Functions: This is a fully managed service that makes it easy to coordinate the components of distributed applications and microservices using visual workflows. By using Step Functions, you can orchestrate services such as AWS Lambda and AWS Glue in a seamless manner. It provides a graphical interface to arrange and visualize the sequence of tasks, which simplifies the management of the workflow.

2. Minimal Management Overhead: Since Step Functions is a fully managed service, it automatically handles the underlying infrastructure, scaling, and error handling. This reduces the operational burden on the data engineer, compared to solutions that require managing EC2 instances or Kubernetes clusters.

3. Integration with AWS Services: Step Functions is natively integrated with many AWS services, including Lambda and Glue. This integration allows you to easily chain together a Lambda function and an AWS Glue job within a state machine, ensuring smooth operation without the need for custom integration code.

4. Reliability and Resilience: Step Functions provides built-in error handling and retry logic, which increases the reliability of the workflow. If the Lambda function or Glue job fails, Step Functions can automatically retry or execute predefined error-handling procedures.

5. Comparison to Other Options: - Option B and D (Apache Airflow on EC2 or EKS): While Apache Airflow is a popular orchestration tool, deploying it on EC2 or EKS involves significant setup and management overhead. You would need to manage the underlying infrastructure, handle scaling, and ensure high availability, which is more complex than using a fully managed service like Step Functions. - Option C (AWS Glue workflow): AWS Glue workflows are specifically designed for managing Glue jobs and might not natively support invoking Lambda functions in the same seamless manner as Step Functions. While possible, it would require additional setup and might not be as straightforward as using Step Functions.

In summary, AWS Step Functions provides an easy-to-use, fully managed, and integrated solution for orchestrating a data pipeline consisting of a Lambda function and a Glue job, minimizing management overhead compared to the alternatives.

Question # 12

Date: June 15, 2024

A retail company uses AWS Glue for extract, transform, and load (ETL) operations on a dataset that contains information about customer orders. The company wants to implement specific validation rules to ensure data accuracy and consistency.

Which solution will meet these requirements?

A. Use AWS Glue job bookmarks to track the data for accuracy and consistency.
B. Create custom AWS Glue Data Quality rulesets to define specific data quality checks.
C. Use the built-in AWS Glue Data Quality transforms for standard data quality validations.
D. Use AWS Glue Data Catalog to maintain a centralized data schema and metadata repository.

Correct Answer:

A. Use AWS Glue job bookmarks to track the data for accuracy and consistency.

B. Create custom AWS Glue Data Quality rulesets to define specific data quality checks.

C. Use the built-in AWS Glue Data Quality transforms for standard data quality validations.

D. Use AWS Glue Data Catalog to maintain a centralized data schema and metadata repository.

Explanation:

The correct answer is B, which suggests creating custom AWS Glue Data Quality rulesets to define specific data quality checks. Let's break down why this is the optimal solution for ensuring data accuracy and consistency during ETL operations with AWS Glue.

AWS Glue is a fully managed ETL service that makes it easy to prepare and transform data. When it comes to data quality, AWS Glue provides a feature known as AWS Glue Data Quality, which allows users to define and enforce data quality rules. This capability is crucial for businesses like a retail company handling customer orders, where data integrity is paramount.

Here's why option B is the best choice:

1. Custom Validation Rules: By creating custom AWS Glue Data Quality rulesets, the company can define specific validation checks tailored to their unique business requirements. This means they can implement rules that are highly specific to their dataset, such as validating order amounts, checking for missing customer information, or ensuring order dates are within expected ranges.

2. Flexibility and Control: Custom rulesets provide flexibility and control over the validation process. The company can specify exactly what constitutes a valid dataset and receive alerts or take corrective actions when data does not meet these criteria.

3. Automation and Integration: AWS Glue allows these data quality checks to be seamlessly integrated into the ETL workflow. This ensures that data quality validation is part of the data processing pipeline, providing automated and consistent checks without additional manual intervention.

4. Scalability: As the dataset grows or evolves, custom rulesets can be updated or expanded to accommodate new validation requirements. This scalability is essential for businesses that expect changes in their data over time.

Option C, which mentions using built-in AWS Glue Data Quality transforms, refers to standard, predefined checks. While useful for general validation, they may not cover all the specific requirements a company might have for its unique dataset, hence the need for custom rulesets.

Option A, using job bookmarks, is more about tracking data processing state to avoid reprocessing the same data rather than ensuring data quality.

Option D, using AWS Glue Data Catalog, is focused on metadata management and schema maintenance, which is important but not directly related to implementing data validation rules.

In summary, creating custom AWS Glue Data Quality rulesets (option B) provides the precise, flexible, and integrated approach needed to enforce specific data validation rules, ensuring data accuracy and consistency tailored to the company's needs.

Question # 13

Date:

A company stores daily records of the financial performance of investment portfolios in .csv format in an Amazon S3 bucket. A data engineer uses AWS Glue crawlers to crawl the S3 data.

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog.

Which solution will meet these requirements?

A. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Create a daily schedule to run the crawler. Configure the output destination to a new path in the existing S3 bucket.
B. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Create a daily schedule to run the crawler. Specify a database name for the output.
C. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Allocate data processing units (DPUs) to run the crawler every day. Specify a database name for the output.
D. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Allocate data processing units (DPUs) to run the crawler every day. Configure the output destination to a new path in the existing S3 bucket.

Correct Answer:

A. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Create a daily schedule to run the crawler. Configure the output destination to a new path in the existing S3 bucket.

B. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Create a daily schedule to run the crawler. Specify a database name for the output.

C. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Allocate data processing units (DPUs) to run the crawler every day. Specify a database name for the output.

D. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Allocate data processing units (DPUs) to run the crawler every day. Configure the output destination to a new path in the existing S3 bucket.

Explanation:

The correct answer is B, and here's why:

1. IAM Role and Policy: - In AWS Glue, the crawler needs permissions to access both the source data in Amazon S3 and to interact with the AWS Glue Data Catalog. The AWSGlueServiceRole policy is specifically designed to give the necessary permissions for AWS Glue operations. This includes accessing S3 data and writing metadata to the Glue Data Catalog. Therefore, using the AWSGlueServiceRole policy is appropriate for this task.

2. Crawler Setup: - The crawler is set up to specify the S3 bucket path where the .csv files are stored. This is essential because the crawler needs to know where to look for the data to catalog it.

3. Scheduled Crawling: - By creating a daily schedule for the crawler, the company ensures that the data in the Glue Data Catalog is updated daily. This is crucial for maintaining current records of the financial performance data, as the requirement is to make the S3 data accessible on a daily basis.

4. Output Specification: - Specifying a database name for the output is necessary for organizing the metadata in the Glue Data Catalog. The Glue Data Catalog is essentially a centralized metadata repository, and organizing it by specifying a database helps in accessing and managing the metadata efficiently.

5. Why Other Options Are Not Suitable: - Option A suggests using the AmazonS3FullAccess policy, which is too broad and not tailored for Glue operations. It also mentions configuring the output to a new S3 path, which is unnecessary since the goal is to update the Glue Data Catalog, not to move or copy the data. - Option C combines AmazonS3FullAccess with specifying a database name but unnecessarily mentions allocating DPUs for the crawler, which is more relevant for AWS Glue jobs, not crawlers. - Option D also involves unnecessary configuration of a new S3 path for output when the task is simply to update the Glue Data Catalog.

In summary, answer B correctly sets up the necessary permissions, specifies the data source and output in the Glue Data Catalog, and ensures that the process is automated daily, aligning perfectly with the requirements.

Question # 14

Date: January 20, 2024

A company loads transaction data for each day into Amazon Redshift tables at the end of each day. The company wants to have the ability to track which tables have been loaded and which tables still need to be loaded.

A data engineer wants to store the load statuses of Redshift tables in an Amazon DynamoDB table. The data engineer creates an AWS Lambda function to publish the details of the load statuses to DynamoDB.

How should the data engineer invoke the Lambda function to write load statuses to the DynamoDB table?

A. Use a second Lambda function to invoke the first Lambda function based on Amazon CloudWatch events.
B. Use the Amazon Redshift Data API to publish an event to Amazon EventBridge. Configure an EventBridge rule to invoke the Lambda function.
C. Use the Amazon Redshift Data API to publish a message to an Amazon Simple Queue Service (Amazon SQS) queue. Configure the SQS queue to invoke the Lambda function.
D. Use a second Lambda function to invoke the first Lambda function based on AWS CloudTrail events.

Correct Answer:

A. Use a second Lambda function to invoke the first Lambda function based on Amazon CloudWatch events.

B. Use the Amazon Redshift Data API to publish an event to Amazon EventBridge. Configure an EventBridge rule to invoke the Lambda function.

C. Use the Amazon Redshift Data API to publish a message to an Amazon Simple Queue Service (Amazon SQS) queue. Configure the SQS queue to invoke the Lambda function.

D. Use a second Lambda function to invoke the first Lambda function based on AWS CloudTrail events.

Explanation:

The correct answer is B: Use the Amazon Redshift Data API to publish an event to Amazon EventBridge. Configure an EventBridge rule to invoke the Lambda function.

Here's why this is the best option:

1. Amazon Redshift Data API: The Amazon Redshift Data API is designed to simplify access to Amazon Redshift by offering an easy way to interact with Redshift data using SQL statements. This API is well-suited for applications where you need to run SQL queries and manage Redshift data programmatically, without the need to maintain a persistent connection.

2. EventBridge Integration: Amazon EventBridge, previously known as CloudWatch Events, is a serverless event bus service that allows you to connect data from different applications using events. It can capture state change events from various AWS services and route them to targets like Lambda functions. This makes it a powerful tool for setting up automated workflows based on events.

3. Event-based Invocation: By using the Redshift Data API to publish an event to EventBridge, you are effectively creating a real-time, event-driven architecture. When the transaction data is loaded into Redshift, an event can be published to EventBridge. This event acts as a trigger.

4. Configuring EventBridge Rule: You can configure an EventBridge rule to listen for specific events, such as the completion of data loading into Redshift. This rule can then be set to invoke the Lambda function, which updates the load status in the DynamoDB table.

5. Decoupled and Scalable Architecture: This approach leverages AWS's managed services to create a decoupled and scalable architecture. The use of EventBridge ensures that the Lambda function is invoked only when necessary, avoiding unnecessary invocations and reducing costs.

In contrast, the other options involve more complexity or are not ideal for this specific use case: - Option A and D suggest using a second Lambda function with CloudWatch or CloudTrail, which adds unnecessary complexity and doesn't directly align with event-driven processing. - Option C suggests using Amazon SQS, which introduces a queue layer that might not be necessary for the direct event-driven invocation required in this scenario.

Overall, Option B provides a streamlined, event-driven solution that efficiently handles the task of updating load statuses in DynamoDB using AWS's serverless services.

Question # 15

Date: January 20, 2024

A data engineer needs to securely transfer 5 TB of data from an on-premises data center to an Amazon S3 bucket. Approximately 5% of the data changes every day. Updates to the data need to be regularly proliferated to the S3 bucket. The data includes files that are in multiple formats. The data engineer needs to automate the transfer process and must schedule the process to run periodically.

Which AWS service should the data engineer use to transfer the data in the MOST operationally efficient way?

A. AWS DataSync

B. AWS Glue
C. AWS Direct Connect
D. Amazon S3 Transfer Acceleration

Correct Answer:

A. AWS DataSync

B. AWS Glue

C. AWS Direct Connect

D. Amazon S3 Transfer Acceleration

Explanation:

AWS DataSync is the correct choice for this scenario because it is specifically designed to automate and accelerate the process of transferring large amounts of data between on-premises storage systems and AWS services like Amazon S3. Here are the reasons why AWS DataSync is the most operationally efficient solution for the given requirements:

1. Secure and Efficient Data Transfer: AWS DataSync provides a secure and efficient way to transfer large volumes of data. It uses a purpose-built protocol that is optimized for high-speed data transfer, which is essential when dealing with large datasets like 5 TB.

2. Handles Data Changes: Since 5% of the data changes every day, DataSync is well-suited for this scenario as it can perform incremental data transfers. This means it will only transfer the changed parts of the data after the initial full transfer, making the process faster and more efficient.

3. Automation and Scheduling: DataSync allows for automated, scheduled data transfers. This is crucial for the requirement to regularly update the S3 bucket with changes from the on-premises data center. It can be set up to run transfers on a schedule that aligns with the frequency of data changes, ensuring that the S3 bucket is always up-to-date.

4. Support for Multiple File Formats: DataSync can handle files in multiple formats without requiring any additional configuration. This flexibility is important given the diverse file formats mentioned in the requirement.

5. Operational Simplicity: DataSync is a fully managed service, which means the data engineer does not need to worry about the underlying infrastructure or manage complex scripts and processes for data transfer. This reduces the operational overhead and allows the data engineer to focus on more strategic tasks.

In contrast, the other options have limitations for this specific use case:

- AWS Glue: This is primarily a data integration service used for data preparation and transformation, not for direct data transfer from on-premises to S3.

- AWS Direct Connect: This is a network service that provides a dedicated connection from on-premises to AWS. While it can be part of a data transfer strategy, it does not inherently automate or schedule data transfers.

- Amazon S3 Transfer Acceleration: This service is designed to speed up transfers over long distances to S3 but is not specifically tailored for ongoing, automated data synchronization from on-premises systems.

Overall, AWS DataSync aligns perfectly with the needs of this scenario by providing a secure, automated, and efficient solution for transferring and synchronizing large datasets to Amazon S3.

Question # 16

Date: February 02, 2024

A data engineer has a one-time task to read data from objects that are in Apache Parquet format in an Amazon S3 bucket. The data engineer needs to query only one column of the data.

Which solution will meet these requirements with the LEAST operational overhead?

A. Configure an AWS Lambda function to load data from the S3 bucket into a pandas dataframe. Write a SQL SELECT statement on the dataframe to query the required column.
B. Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.
C. Prepare an AWS Glue DataBrew project to consume the S3 objects and to query the required column.
D. Run an AWS Glue crawler on the S3 objects. Use a SQL SELECT statement in Amazon Athena to query the required column.

Correct Answer:

A. Configure an AWS Lambda function to load data from the S3 bucket into a pandas dataframe. Write a SQL SELECT statement on the dataframe to query the required column.

B. Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.

C. Prepare an AWS Glue DataBrew project to consume the S3 objects and to query the required column.

D. Run an AWS Glue crawler on the S3 objects. Use a SQL SELECT statement in Amazon Athena to query the required column.

Explanation:

The correct answer is B: Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.

Here's why this is the best choice:

1. Understanding S3 Select: S3 Select is an Amazon Simple Storage Service (S3) feature that enables users to retrieve a subset of data from an object by using SQL expressions. This is particularly useful when you need to extract specific data without transferring the entire object over the network, which can save time and reduce costs.

2. Operational Overhead: The question specifically emphasizes minimizing operational overhead. S3 Select is designed to be simple and efficient for exactly this type of task. It allows you to directly query data within an S3 object without the need for additional infrastructure or processing steps, which reduces complexity and operational effort.

3. Suitability for the Task: Since the data is in Apache Parquet format, S3 Select can efficiently read and filter the data using SQL. Parquet is a columnar storage format, which means it's optimized for scenarios where you need to retrieve specific columns. S3 Select leverages this by reading only the required column, further reducing the amount of data processed and transferred.

4. Comparison with Other Options: - Option A involves configuring an AWS Lambda function, which introduces additional management overhead. You need to write, deploy, and possibly troubleshoot the Lambda function, along with managing any necessary permissions and execution roles. - Option C suggests using AWS Glue DataBrew, which is more suited for data cleaning and preparation tasks. Setting up a DataBrew project involves more setup and configuration, which isn't necessary for a one-time, simple column retrieval task. - Option D involves running an AWS Glue crawler and using Amazon Athena. While Athena is a powerful tool for querying data in S3, setting up a Glue crawler to catalog the data adds unnecessary steps and complexity compared to directly querying with S3 Select.

In summary, S3 Select provides a direct, efficient, and low-overhead way to query specific columns from Parquet files stored in S3, making it the ideal solution for this one-time task.

Question # 17

Date: February 02, 2024

A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisioned capacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Amazon Athena to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.

B. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Redshift Spectrum to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.
C. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format. Store the transformed data in an S3 bucket. Use Amazon Athena to query the original and transformed data from the S3 bucket.
D. Use AWS Lake Formation to create a data lake. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format. Store the transformed data in an S3 bucket. Use Amazon Athena or Redshift Spectrum to query the data.

Correct Answer:

A. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Amazon Athena to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.

B. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Redshift Spectrum to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.

C. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format. Store the transformed data in an S3 bucket. Use Amazon Athena to query the original and transformed data from the S3 bucket.

D. Use AWS Lake Formation to create a data lake. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format. Store the transformed data in an S3 bucket. Use Amazon Athena or Redshift Spectrum to query the data.

Explanation:

The question is about finding a solution that allows data scientists to query multiple data sources using SQL-like syntax while minimizing operational overhead. Let's break down why option A is the correct choice.

1. AWS Glue and Glue Data Catalog: AWS Glue is a managed ETL (Extract, Transform, Load) service that makes it easy to prepare and transform data for analytics. Using AWS Glue to crawl the data sources (both JSON and CSV formats stored in Amazon S3, and other data sources like RDS and DynamoDB) allows the data to be cataloged efficiently. The AWS Glue Data Catalog serves as a central metadata repository, which is essential for querying data using different AWS analytics services.

2. Amazon Athena: Athena is an interactive query service that allows you to use standard SQL to query data directly in Amazon S3. It is serverless, meaning there is no need to manage any infrastructure, which significantly reduces operational overhead. Athena can query both structured data (like CSV) and semi-structured data (like JSON) stored in S3. For structured data sources, SQL can be used directly.

3. PartiQL: PartiQL is a SQL-compatible query language that extends SQL to work with nested data formats such as JSON. With PartiQL, you can query and manipulate JSON data in a way that feels familiar to SQL users. This is particularly useful for querying the JSON datasets in S3, allowing you to handle both structured and semi-structured data using similar syntax.

4. Operational Overhead: The solution minimizes operational overhead by leveraging managed services (AWS Glue and Amazon Athena), which are designed to handle infrastructure, scalability, and maintenance tasks automatically. This way, the data engineering team can focus more on developing queries and insights rather than managing servers or data pipelines.

In summary, option A provides a cohesive solution that allows querying across different data formats with minimal setup and maintenance. The use of AWS Glue for metadata management and Amazon Athena for querying, along with PartiQL for JSON, ensures that the data scientists can work with all types of data using familiar SQL syntax while the operational overhead is kept low.

Question # 18

Date: January 20, 2024

A company uses an Amazon Redshift cluster that runs on RA3 nodes. The company wants to scale read and write capacity to meet demand. A data engineer needs to identify a solution that will turn on concurrency scaling.

Which solution will meet this requirement?

A. Turn on concurrency scaling in workload management (WLM) for Redshift Serverless workgroups.
B. Turn on concurrency scaling at the workload management (WLM) queue level in the Redshift cluster.
C. Turn on concurrency scaling in the settings during the creation of any new Redshift cluster.
D. Turn on concurrency scaling for the daily usage quota for the Redshift cluster.

Correct Answer:

A. Turn on concurrency scaling in workload management (WLM) for Redshift Serverless workgroups.

B. Turn on concurrency scaling at the workload management (WLM) queue level in the Redshift cluster.

C. Turn on concurrency scaling in the settings during the creation of any new Redshift cluster.

D. Turn on concurrency scaling for the daily usage quota for the Redshift cluster.

Explanation:

The correct answer is B: Turn on concurrency scaling at the workload management (WLM) queue level in the Redshift cluster.

Here's why this is the correct choice:

1. Understanding Concurrency Scaling in Redshift: Concurrency scaling in Amazon Redshift is designed to handle periods of high demand by automatically adding additional query processing power when you need it. This feature ensures that your queries can run without delay, even when the workload is high, by spinning up additional clusters to handle the increased load.

2. RA3 Nodes and Concurrency Scaling: The RA3 node type in Amazon Redshift supports concurrency scaling. This is important because not all node types in Redshift support this feature. RA3 nodes are specifically designed to separate storage and compute, allowing for more flexible and scalable operations, including concurrency scaling.

3. Workload Management (WLM) in Redshift: WLM is a feature in Amazon Redshift that allows you to define how queries are managed and prioritized. By configuring WLM, you can allocate resources to different types of workloads, ensuring that critical queries have the resources they need to execute efficiently.

4. Why Option B is Correct: Concurrency scaling needs to be enabled at the WLM queue level within a Redshift cluster. By doing so, you allow specific queues to benefit from additional resources during peak times. This means that when the workload in a particular queue exceeds the allocated capacity, Redshift can automatically scale out to additional clusters to handle the increased demand.

5. Why Other Options are Incorrect: - Option A: This option refers to Redshift Serverless, which is not relevant here as the question specifies a Redshift cluster running on RA3 nodes, not a serverless configuration. - Option C: Concurrency scaling is not something you simply turn on at the creation of a new cluster; it specifically needs to be configured at the WLM queue level within the cluster. - Option D: This is not an accurate representation of how concurrency scaling is enabled. While there are usage quotas for concurrency scaling (since it incurs additional costs), the feature itself is enabled at the WLM queue level, not through daily usage quotas.

In summary, option B accurately reflects the necessary steps to enable concurrency scaling for a Redshift cluster running on RA3 nodes by configuring it at the workload management (WLM) queue level. This setup allows the cluster to dynamically adjust resources in response to varying query loads, ensuring efficient processing and scaling of read and write operations.

Question # 19

Date: February 02, 2024

A company stores data in a data lake that is in Amazon S3. Some data that the company stores in the data lake contains personally identifiable information (PII). Multiple user groups need to access the raw data. The company must ensure that user groups can access only the PII that they require.

Which solution will meet these requirements with the LEAST effort?

A. Use Amazon Athena to query the data. Set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. Assign each user to the IAM role that matches the user's PII access requirements.
B. Use Amazon QuickSight to access the data. Use column-level security features in QuickSight to limit the PII that users can retrieve from Amazon S3 by using Amazon Athena. Define QuickSight access levels based on the PII access requirements of the users.
C. Build a custom query builder UI that will run Athena queries in the background to access the data. Create user groups in Amazon Cognito. Assign access levels to the user groups based on the PII access requirements of the users.
D. Create IAM roles that have different levels of granular access. Assign the IAM roles to IAM user groups. Use an identity-based policy to assign access levels to user groups at the column level.

Correct Answer:

A. Use Amazon Athena to query the data. Set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. Assign each user to the IAM role that matches the user's PII access requirements.

B. Use Amazon QuickSight to access the data. Use column-level security features in QuickSight to limit the PII that users can retrieve from Amazon S3 by using Amazon Athena. Define QuickSight access levels based on the PII access requirements of the users.

C. Build a custom query builder UI that will run Athena queries in the background to access the data. Create user groups in Amazon Cognito. Assign access levels to the user groups based on the PII access requirements of the users.

D. Create IAM roles that have different levels of granular access. Assign the IAM roles to IAM user groups. Use an identity-based policy to assign access levels to user groups at the column level.

Explanation:

The correct answer is A: Use Amazon Athena to query the data. Set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. Assign each user to the IAM role that matches the user's PII access requirements.

Here's why this solution is the most effective with the least effort:

1. Amazon Athena: This is a serverless interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. It's a cost-effective solution because it doesn't require infrastructure provisioning or maintenance.

2. AWS Lake Formation: This service simplifies the process of setting up a secure data lake. It allows you to define and enforce fine-grained access control policies for data stored in S3. With Lake Formation, you can create data filters that define which parts of the data different user groups can access, particularly useful for managing access to sensitive data like PII.

3. Data Filters and Access Control: By using Lake Formation, you can create data filters that determine the specific PII data that each IAM role can access. This feature enables you to establish precise access levels without manually managing permissions for each dataset or user, which simplifies administration.

4. IAM Roles: Assigning IAM roles to users based on their access requirements is a scalable and manageable approach. By associating each user with the appropriate IAM role, you ensure they only access the data they are authorized to see without needing to define separate access policies for each user.

5. Least Effort: Compared to the other options, this approach minimizes the need for custom development and complex configurations. It leverages existing AWS services designed to work together, reducing the need for additional tools or custom UI development, as mentioned in other options.

Overall, this solution effectively balances security, simplicity, and scalability, making it the most efficient way to manage access to PII within a data lake stored in Amazon S3.

Question # 20

Date: January 20, 2024

A company is migrating on-premises workloads to AWS. The company wants to reduce overall operational overhead. The company also wants to explore serverless options.

The company's current workloads use Apache Pig, Apache Oozie, Apache Spark, Apache Hbase, and Apache Flink. The on-premises workloads process petabytes of data in seconds. The company must maintain similar or better performance after the migration to AWS.

Which extract, transform, and load (ETL) service will meet these requirements?

A. AWS Glue
B. Amazon EMR
C. AWS Lambda
D. Amazon Redshift

Correct Answer:

A. AWS Glue

B. Amazon EMR

C. AWS Lambda

D. Amazon Redshift

Explanation:

The correct answer to this question is B: Amazon EMR.

Let's break down why Amazon EMR is the most suitable choice for the company's requirements:

1. Compatibility with Existing Tools: The company's current on-premises workloads use Apache Pig, Apache Oozie, Apache Spark, Apache HBase, and Apache Flink. Amazon EMR is a managed cluster platform that simplifies running big data frameworks, including the exact tools the company is using. This means that the company can easily migrate these workloads to AWS with minimal changes to their existing processes.

2. Scalability and Performance: Amazon EMR is designed to handle vast amounts of data quickly and efficiently, which aligns with the company's requirement to process petabytes of data in seconds. EMR can scale out to thousands of compute instances and can be tuned to optimize for both performance and cost.

3. Reduced Operational Overhead: While Amazon EMR is not serverless, it significantly reduces operational overhead compared to managing these frameworks on-premises. It automates provisioning, configuration, and tuning of clusters, enabling the company to focus more on data processing rather than infrastructure management.

4. Serverless Consideration: Although the company is interested in exploring serverless options, their specific need to run tools like Apache Spark and Flink—traditionally not serverless—makes Amazon EMR a more practical choice. AWS does provide serverless options like AWS Glue, but Glue is primarily designed for ETL operations and may not support all the specific frameworks and use cases as directly as EMR does.

5. Performance Requirements: Maintaining or improving performance is crucial for the company. Amazon EMR can be optimized for high-performance data processing, and its ability to integrate with AWS's extensive suite of services allows for enhanced data processing capabilities.

In contrast, other options like AWS Glue, AWS Lambda, and Amazon Redshift don't fully align with the requirements:

- AWS Glue: It's a serverless ETL service but doesn't natively support all the frameworks like Apache Spark, HBase, and Flink to the extent that EMR does. It's more suited for simpler ETL tasks.

- AWS Lambda: While serverless, it's not intended for running complex big data frameworks due to its limitations on execution time and resources.

- Amazon Redshift: Primarily a data warehouse service, it wouldn't directly support the execution of the company's existing big data frameworks and workloads.

Therefore, Amazon EMR is the best fit for the company's needs, offering a managed solution that supports their existing tools, meets performance requirements, and reduces operational complexity.

Question # 21

Date: February 01, 2024

A data engineer must use AWS services to ingest a dataset into an Amazon S3 data lake. The data engineer profiles the dataset and discovers that the dataset contains personally identifiable information (PII). The data engineer must implement a solution to profile the dataset and obfuscate the PII.

Which solution will meet this requirement with the LEAST operational effort?

A. Use an Amazon Kinesis Data Firehose delivery stream to process the dataset. Create an AWS Lambda transform function to identify the PII. Use an AWS SDK to obfuscate the PII. Set the S3 data lake as the target for the delivery stream.
B. Use the Detect PII transform in AWS Glue Studio to identify the PII. Obfuscate the PII. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
C. Use the Detect PII transform in AWS Glue Studio to identify the PII. Create a rule in AWS Glue Data Quality to obfuscate the PII. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
D. Ingest the dataset into Amazon DynamoDB. Create an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data. Use the same Lambda function to ingest the data into the S3 data lake.

Correct Answer:

A. Use an Amazon Kinesis Data Firehose delivery stream to process the dataset. Create an AWS Lambda transform function to identify the PII. Use an AWS SDK to obfuscate the PII. Set the S3 data lake as the target for the delivery stream.

B. Use the Detect PII transform in AWS Glue Studio to identify the PII. Obfuscate the PII. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.

C. Use the Detect PII transform in AWS Glue Studio to identify the PII. Create a rule in AWS Glue Data Quality to obfuscate the PII. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.

D. Ingest the dataset into Amazon DynamoDB. Create an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data. Use the same Lambda function to ingest the data into the S3 data lake.

Explanation:

The correct answer is B, and here's why this option is the most suitable choice with the least operational effort:

1. AWS Glue Studio with Detect PII Transform: - AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy to prepare and load data for analytics. It offers a specific feature called the "Detect PII" transform, which is designed to identify personally identifiable information in datasets. By using this feature, the data engineer can efficiently detect PII without having to develop custom code or logic to handle this identification, which reduces the operational effort significantly.

2. Obfuscation in AWS Glue: - Once the PII is identified using the Detect PII transform, the same AWS Glue environment can be used to obfuscate this data. This could involve techniques such as hashing, masking, or encrypting the PII fields, all of which can be configured within Glue. This integrated approach within a single service streamlines the process of PII management.

3. AWS Step Functions for Orchestration: - AWS Step Functions is a serverless function orchestrator that makes it easy to coordinate multiple AWS services into serverless workflows. By using Step Functions, the data engineer can set up a robust pipeline that automates the entire process of data ingestion, PII detection, and obfuscation. This orchestration minimizes manual intervention and reduces operational overhead since Step Functions handle the sequential flow of tasks automatically.

4. Ingestion into Amazon S3: - After the PII has been detected and obfuscated, the final step in the pipeline involves ingesting the cleaned data into the Amazon S3 data lake. S3 is a highly scalable and durable storage service that acts as an ideal data lake destination. The use of Step Functions ensures that this step is seamlessly integrated into the overall data processing workflow.

Choosing option B leverages AWS Glue's built-in capabilities for PII detection and obfuscation, combined with AWS Step Functions' orchestration power, to deliver a solution that is efficient and requires minimal manual intervention. The other options involve more complex and fragmented approaches that would require more custom development and operational management, thus increasing the effort needed.

Question # 22 Date:

A company maintains multiple extract, transform, and load (ETL) workflows that ingest data from the company's operational databases into an Amazon S3 based data lake. The ETL workflows use AWS Glue and Amazon EMR to process data.

The company wants to improve the existing architecture to provide automated orchestration and to require minimal manual effort.

Which solution will meet these requirements with the LEAST operational overhead?

A. AWS Glue workflows
B. AWS Step Functions tasks
C. AWS Lambda functions
D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) workflows

Correct Answer:

A. AWS Glue workflows

B. AWS Step Functions tasks

C. AWS Lambda functions

D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) workflows

Explanation:

Certainly! Let's break down why option B, AWS Step Functions tasks, is the correct choice for improving the company's ETL workflows with the least operational overhead.

1. Automation and Orchestration: AWS Step Functions is a fully managed service that makes it easy to coordinate the components of distributed applications and microservices using visual workflows. It excels in automating complex workflows by handling the sequence, retry logic, and parallel processing of tasks. For ETL processes, Step Functions can orchestrate the different steps involved—such as data extraction, transformation, and loading—by linking AWS Glue and Amazon EMR tasks into a cohesive flow.

2. Integration with AWS Services: Step Functions integrates seamlessly with other AWS services, including AWS Glue and Amazon EMR. This integration allows you to easily manage and automate the ETL processes without needing to write custom code to handle the orchestration. You can trigger Glue jobs and manage EMR clusters directly from your Step Functions state machine, ensuring that each step in your ETL process is executed in the correct order.

3. Minimal Operational Overhead: Since Step Functions is a fully managed service, it significantly reduces the operational overhead compared to setting up and maintaining orchestration infrastructure yourself. You don't need to worry about servers, scaling, or complex error handling since Step Functions manages these aspects for you. This means less manual intervention and maintenance, aligning perfectly with the company's goal of minimal manual effort.

4. Reliability and Error Handling: Step Functions provides built-in error handling, retry capabilities, and state management, which are crucial for ensuring that ETL workflows run smoothly and can recover from failures automatically. This reduces the need for manual monitoring and intervention, further decreasing operational overhead.

5. Cost-effectiveness: Using Step Functions can be more cost-effective than other options, especially when you consider the time and resources saved from reduced manual management and the elimination of the need to run your own orchestration infrastructure.

While other options like AWS Glue workflows or Amazon MWAA could also automate ETL processes, they may involve more setup complexity or management overhead. AWS Lambda functions, on the other hand, are more suited for executing code in response to events rather than orchestrating complex workflows.

In summary, AWS Step Functions tasks offer a robust, low-maintenance solution for orchestrating ETL workflows, making it the ideal choice for the company's requirements of automation with minimal manual effort.

Question # 23

Date: February 01, 2024

A company currently stores all of its data in Amazon S3 by using the S3 Standard storage class.

A data engineer examined data access patterns to identify trends. During the first 6 months, most data files are accessed several times each day. Between 6 months and 2 years, most data files are accessed once or twice each month. After 2 years, data files are accessed only once or twice each year.

The data engineer needs to use an S3 Lifecycle policy to develop new data storage rules. The new storage solution must continue to provide high availability.

Which solution will meet these requirements in the MOST cost-effective way?

A. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.
B. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.
C. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. Transfer objects to S3 Glacier Deep Archive after 2 years.
D. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. Transfer objects to S3 Glacier Deep Archive after 2 years.

Correct Answer:

A. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.

B. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.

C. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. Transfer objects to S3 Glacier Deep Archive after 2 years.

D. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. Transfer objects to S3 Glacier Deep Archive after 2 years.

Explanation:

The correct answer is B, and here's why:

When designing a cost-effective storage solution using Amazon S3, it's important to balance between cost and access frequency, while ensuring high availability. Let's break down the components of the solution:

1. Initial Storage in S3 Standard: - S3 Standard is ideal for frequently accessed data. It provides high durability, availability, and performance, which suits the initial phase where data is accessed several times daily.

2. After 6 Months - Transition to S3 Standard-Infrequent Access (S3 Standard-IA): - Once data access decreases to once or twice each month, transitioning to S3 Standard-IA becomes cost-effective. This storage class is designed for data that is accessed less frequently but still requires rapid access when needed. It offers the same high durability and availability as S3 Standard but at a lower cost, making it suitable for this intermediate access frequency.

3. After 2 Years - Transition to S3 Glacier Flexible Retrieval: - For data accessed just once or twice a year, S3 Glacier Flexible Retrieval is a more cost-effective solution. It is designed for archival storage where occasional access is needed, offering significant cost savings compared to more frequently accessed storage classes. - While this storage class provides slightly longer retrieval times compared to S3 Standard-IA, it's still capable of providing access within hours, which is generally acceptable for archival data.

Why Not the Other Options?

- Option A: S3 One Zone-IA is less expensive than S3 Standard-IA but offers lower availability and durability because it stores data in a single availability zone. Given the requirement for high availability, this makes it less suitable. - Options C and D: S3 Glacier Deep Archive offers even lower storage costs than S3 Glacier Flexible Retrieval, but with significantly longer retrieval times. It's typically used for data that is rarely accessed, where retrieval time is not a critical factor, which might not be suitable if the data needs to be accessed once or twice a year with more urgency.

In summary, the chosen solution (Option B) balances cost-effectiveness with the need for high availability and relatively quick access, aligning with the specified data access patterns and requirements.

Question # 24 Date:

A company maintains an Amazon Redshift provisioned cluster that the company uses for extract, transform, and load (ETL) operations to support critical analysis tasks. A sales team within the company maintains a Redshift cluster that the sales team uses for business intelligence (BI) tasks.

The sales team recently requested access to the data that is in the ETL Redshift cluster so the team can perform weekly summary analysis tasks. The sales team needs to join data from the ETL cluster with data that is in the sales team's BI cluster.

The company needs a solution that will share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution must minimize usage of the computing resources of the ETL cluster.

Which solution will meet these requirements?

A. Set up the sales team BI cluster as a consumer of the ETL cluster by using Redshift data sharing.
B. Create materialized views based on the sales team's requirements. Grant the sales team direct access to the ETL cluster.
C. Create database views based on the sales team's requirements. Grant the sales team direct access to the ETL cluster.
D. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every week. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

Correct Answer:

A. Set up the sales team BI cluster as a consumer of the ETL cluster by using Redshift data sharing.

B. Create materialized views based on the sales team's requirements. Grant the sales team direct access to the ETL cluster.

C. Create database views based on the sales team's requirements. Grant the sales team direct access to the ETL cluster.

D. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every week. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

Explanation:

The correct answer is A: Set up the sales team BI cluster as a consumer of the ETL cluster by using Redshift data sharing.

Here's why this solution is the best fit for the requirements:

1. Redshift Data Sharing: Amazon Redshift data sharing allows you to share live data securely and in real-time across different Redshift clusters without the need to copy the data. This means that the sales team can access the data in the ETL cluster without duplicating it, which helps in minimizing the usage of computing resources on the ETL cluster. Data sharing is designed to allow multiple clusters to work with the same data seamlessly, making it ideal for scenarios where different teams or departments need access to the same datasets.

2. Performance and Resource Efficiency: By using data sharing, the computational load on the ETL cluster is minimized because the sales team’s BI cluster can directly consume the data. The ETL cluster doesn’t need to perform additional operations like copying or transforming data specifically for the sales team’s access. This ensures that the critical analysis tasks running on the ETL cluster are not interrupted or slowed down due to extra processing for data access.

3. No Data Duplication: Data sharing eliminates the need to create and manage copies of the data. This reduces storage costs and the complexity of ensuring data consistency across multiple clusters. The sales team accesses the most up-to-date data directly from the ETL cluster, which is crucial for accurate analysis.

4. Ease of Use and Management: Setting up data sharing is straightforward and doesn't require complex configurations or additional infrastructure like data transfers to Amazon S3. This makes it a less error-prone and more maintainable solution compared to other options.

In contrast, the other options involve either duplicating data, which increases storage and maintenance overhead (Option D), or granting direct access to the ETL cluster with potential impacts on performance (Options B and C). These alternatives either complicate the architecture or fail to effectively minimize the ETL cluster's resource usage, which is a key requirement for the solution.

Question # 25

Date: August 09, 2024

A company reads data from customer databases that run on Amazon RDS. The databases contain many inconsistent fields. For example, a customer record field that iPnamed place_id in one database is named location_id in another database. The company needs to link customer records across different databases, even when customer record fields do not match.

Which solution will meet these requirements with the LEAST operational overhead?

A. Create a provisioned Amazon EMR cluster to process and analyze data in the databases. Connect to the Apache Zeppelin notebook. Use the FindMatches transform to find duplicate records in the data.
B. Create an AWS Glue crawler to craw the databases. Use the FindMatches transform to find duplicate records in the data. Evaluate and tune the transform by evaluating the performance and results.
C. Create an AWS Glue crawler to craw the databases. Use Amazon SageMaker to construct Apache Spark ML pipelines to find duplicate records in the data.
D. Create a provisioned Amazon EMR cluster to process and analyze data in the databases. Connect to the Apache Zeppelin notebook. Use an Apache Spark ML model to find duplicate records in the data. Evaluate and tune the model by evaluating the performance and results.

Correct Answer:

A. Create a provisioned Amazon EMR cluster to process and analyze data in the databases. Connect to the Apache Zeppelin notebook. Use the FindMatches transform to find duplicate records in the data.

B. Create an AWS Glue crawler to craw the databases. Use the FindMatches transform to find duplicate records in the data. Evaluate and tune the transform by evaluating the performance and results.

C. Create an AWS Glue crawler to craw the databases. Use Amazon SageMaker to construct Apache Spark ML pipelines to find duplicate records in the data.

D. Create a provisioned Amazon EMR cluster to process and analyze data in the databases. Connect to the Apache Zeppelin notebook. Use an Apache Spark ML model to find duplicate records in the data. Evaluate and tune the model by evaluating the performance and results.

Explanation:

The correct answer is B, and here's why:

When faced with the task of linking customer records across different databases with inconsistent fields, you need a solution that minimizes operational overhead while effectively handling the data discrepancies.

1. AWS Glue Crawler: AWS Glue is a fully managed ETL (Extract, Transform, Load) service that makes it easy to prepare your data for analytics. By using an AWS Glue crawler, you can automatically discover and catalog metadata about your data stores. This automation reduces the need for manual intervention, thus lowering operational overhead.

2. FindMatches Transform: AWS Glue includes a feature called FindMatches, which utilizes machine learning to identify duplicate records within your data. This is particularly useful when dealing with inconsistent field names, as it can learn the patterns and relationships between different datasets to accurately link related records. The FindMatches transform is specifically designed for this type of record linkage task, making it a suitable choice for this scenario.

3. Least Operational Overhead: Among the options, using AWS Glue with the FindMatches transform offers the least operational overhead because it provides a serverless architecture. This means you don't have to manage infrastructure, such as provisioning and maintaining Amazon EMR clusters or managing Spark ML pipelines, which are necessary in other options. AWS Glue automates much of the ETL process, which simplifies the data preparation and analysis workflow.

4. Evaluation and Tuning: The FindMatches transform allows for evaluation and tuning based on the performance and results. This feature ensures that you can iteratively improve the accuracy of the record matching process without needing extensive manual effort or infrastructure management.

In contrast, options A, C, and D involve using Amazon EMR or Amazon SageMaker, which typically require more hands-on management of the compute resources and ML model development. This increases the complexity and operational overhead compared to the serverless and automated nature of AWS Glue with FindMatches.

Thus, option B provides a solution that balances effectiveness in linking records with minimal operational overhead, making it the best choice for the given requirements.

Question # 26

Date:

A company is planning to use a provisioned Amazon EMR cluster that runs Apache Spark jobs to perform big data analysis. The company requires high reliability. A big data team must follow best practices for running cost-optimized and long-running workloads on Amazon EMR. The team must find a solution that will maintain the company's current level of performance.

Which combination of resources will meet these requirements MOST cost-effectively? (Choose two.)

A. Use Hadoop Distributed File System (HDFS) as a persistent data store.
B. Use Amazon S3 as a persistent data store.
C. Use x86-based instances for core nodes and task nodes.
D. Use Graviton instances for core nodes and task nodes.
E. Use Spot Instances for all primary nodes.

Correct Answer:

A. Use Hadoop Distributed File System (HDFS) as a persistent data store.

B. Use Amazon S3 as a persistent data store.

C. Use x86-based instances for core nodes and task nodes.

D. Use Graviton instances for core nodes and task nodes.

E. Use Spot Instances for all primary nodes.

Explanation:

Certainly! Let's break down why options B and D are the most cost-effective choices for running a reliable, cost-optimized, and long-running workload on Amazon EMR:

Option B: Use Amazon S3 as a persistent data store.

1. Cost-Effectiveness: Amazon S3 is a highly cost-effective storage solution compared to HDFS on the cluster's local storage. By using Amazon S3, you decrease the need for expensive, high-capacity storage on your EC2 instances, which can significantly reduce costs.

2. Scalability and Durability: S3 offers virtually unlimited storage with high durability (99.999999999% durability). This means you can scale up your storage needs without worrying about hardware constraints, and your data is highly protected against loss.

3. Separation of Storage and Compute: By decoupling storage and compute, you can scale your compute resources (like EMR clusters) independently of your storage needs. This flexibility allows you to optimize costs as you only pay for the compute resources when they are actively used.

4. Ease of Data Management: Using S3 as a persistent store simplifies data management tasks like data backup and archiving, as well as integration with other AWS services.

Option D: Use Graviton instances for core nodes and task nodes.

1. Cost-Performance Benefits: AWS Graviton instances, powered by AWS-designed Arm-based processors, provide significant cost savings (up to 20-40% lower cost) and performance improvements over traditional x86-based instances. This makes them an excellent choice for running compute-intensive workloads, such as big data processing with Apache Spark.

2. Energy Efficiency: Graviton processors are designed for high performance and energy efficiency, which can further reduce operational costs by lowering power consumption and cooling requirements.

3. Compatibility: EMR and Apache Spark have been optimized to run on Graviton instances, ensuring that you can achieve the desired level of performance without compromising on reliability or compatibility.

In summary, using Amazon S3 as a persistent data store provides a highly durable, scalable, and cost-effective solution for managing big data storage, while Graviton instances offer a cost-efficient and high-performance compute option for running Spark jobs on EMR. Together, these choices align with best practices for optimizing costs while maintaining high reliability and performance for long-running workloads on AWS.

Question # 27

Date: February 02, 2024

A company extracts approximately 1 TB of data every day from data sources such as SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. Some of the data sources have undefined data schemas or data schemas that change.

A data engineer must implement a solution that can detect the schema for these data sources. The solution must extract, transform, and load the data to an Amazon S3 bucket. The company has a service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon EMR to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.
B. Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.
C. Create a PySpark program in AWS Lambda to extract, transform, and load the data into the S3 bucket.
D. Create a stored procedure in Amazon Redshift to detect the schema and to extract, transform, and load the data into a Redshift Spectrum table. Access the table from Amazon S3.

Correct Answer:

A. Use Amazon EMR to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.

B. Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.

C. Create a PySpark program in AWS Lambda to extract, transform, and load the data into the S3 bucket.

D. Create a stored procedure in Amazon Redshift to detect the schema and to extract, transform, and load the data into a Redshift Spectrum table. Access the table from Amazon S3.

Explanation:

The correct answer is B: Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.

Here's why this solution is appropriate:

1. AWS Glue Overview: AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy to prepare and load data for analytics. It is designed to handle scenarios where data schemas are undefined or change over time. AWS Glue can automatically discover and interpret the schema from various data sources, which fits perfectly with the company's needs as described in the question.

2. Schema Detection: One of the key features of AWS Glue is its ability to automatically detect and manage schema changes. This is crucial for the company's data sources, which include SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB, especially when some have changing or undefined schemas.

3. Integration with Amazon S3: AWS Glue can directly integrate with Amazon S3, allowing seamless data transfer into S3 buckets. This is aligned with the requirement to load data into S3 quickly and efficiently.

4. Operational Overhead: AWS Glue is a serverless service, meaning it automatically scales to meet demand and does not require the user to manage underlying infrastructure. This results in minimal operational overhead compared to managing a cluster with Amazon EMR or setting up manual pipelines.

5. Apache Spark: AWS Glue uses Apache Spark under the hood, which is a powerful tool for processing large datasets quickly. This capability helps meet the company's service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation. Glue's serverless Spark environment allows tasks to be executed quickly without the need for provisioning resources manually.

6. Alternative Solutions: - Option A (Amazon EMR with Apache Spark) would require more operational management, including setting up and maintaining clusters, which increases overhead. - Option C (AWS Lambda with PySpark) is not practical for processing 1 TB of data daily due to Lambda's execution time limits and memory constraints. - Option D (Amazon Redshift) is designed for data warehousing rather than ETL and schema detection, making it an unfit choice for this scenario, especially given the requirement for rapid, flexible data ingestion.

Overall, AWS Glue provides the right combination of automatic schema detection, integration with Amazon S3, and minimal operational overhead to meet the company's requirements efficiently.

Question # 28

Date: February 03, 2024

A company uses an Amazon QuickSight dashboard to monitor usage of one of the company's applications. The company uses AWS Glue jobs to process data for the dashboard. The company stores the data in a single Amazon S3 bucket. The company adds new data every day.

A data engineer discovers that dashboard queries are becoming slower over time. The data engineer determines that the root cause of the slowing queries is long-running AWS Glue jobs.

Which actions should the data engineer take to improve the performance of the AWS Glue jobs? (Choose two.)

A. Partition the data that is in the S3 bucket. Organize the data by year, month, and day.
B. Increase the AWS Glue instance size by scaling up the worker type.
C. Convert the AWS Glue schema to the DynamicFrame schema class.
D. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.
E. Modify the IAM role that grants access to AWS glue to grant access to all S3 features.

Correct Answer:

A. Partition the data that is in the S3 bucket. Organize the data by year, month, and day.

B. Increase the AWS Glue instance size by scaling up the worker type.

C. Convert the AWS Glue schema to the DynamicFrame schema class.

D. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.

E. Modify the IAM role that grants access to AWS glue to grant access to all S3 features.

Explanation:

Certainly! Let's dive into why options A and B are the correct choices for improving the performance of AWS Glue jobs in this scenario.

Option A: Partition the data that is in the S3 bucket. Organize the data by year, month, and day.

Partitioning the data in your S3 bucket is a widely used technique to improve query performance. When data is partitioned, each partition acts as a subset of the data. This organization allows AWS Glue jobs, as well as any query engines (like Amazon Athena or Amazon Redshift Spectrum) that consume this data, to read only the necessary partitions instead of scanning the entire dataset. By organizing data by year, month, and day, you significantly reduce the amount of data that needs to be processed, which can make your queries much faster. This is especially beneficial when dealing with large datasets, which is likely the case here since the data is growing daily.

Option B: Increase the AWS Glue instance size by scaling up the worker type.

AWS Glue jobs are executed using worker nodes, and the performance of these jobs can be impacted by the resources allocated to them. By increasing the instance size (or scaling up the worker type), you are providing more CPU, memory, and network resources to the job. This can help in processing data more efficiently and reducing the time taken by long-running jobs. If the current Glue worker type is not sufficient to handle the workload, this scaling up can lead to noticeable performance improvements.

Now, let's briefly understand why the other options are not suitable:

Option C: Convert the AWS Glue schema to the DynamicFrame schema class.

While converting data to a DynamicFrame can offer flexibility in handling data transformations, it does not inherently improve performance. In fact, DynamicFrames are specifically designed for handling semi-structured data and might add overhead if the benefits of using them are not required for the specific operations.

Option D: Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.

Reducing the frequency of running jobs does not address the performance issue directly. It might reduce the number of times data is processed, but it will not make each job run faster. In this scenario, the problem is related to the long-running nature of the jobs themselves, not how often they are executed.

Option E: Modify the IAM role that grants access to AWS Glue to grant access to all S3 features.

IAM roles and permissions primarily control access, not performance. Granting broader S3 access will not improve the performance of AWS Glue jobs. It's important to follow the principle of least privilege and provide only necessary permissions, which is unrelated to the performance issues being faced.

In summary, partitioning the data (Option A) and scaling up the AWS Glue instance size (Option B) are practical measures to tackle the performance slowdown of AWS Glue jobs by reducing the amount of data processed and increasing computing resources, respectively.

Question # 29

Date:

A data engineer runs Amazon Athena queries on data that is in an Amazon S3 bucket. The Athena queries use AWS Glue Data Catalog as a metadata table.

The data engineer notices that the Athena query plans are experiencing a performance bottleneck. The data engineer determines that the cause of the performance bottleneck is the large number of partitions that are in the S3 bucket. The data engineer must resolve the performance bottleneck and reduce Athena query planning time.

Which solutions will meet these requirements? (Choose two.)

A. Create an AWS Glue partition index. Enable partition filtering.
B. Bucket the data based on a column that the data have in common in a WHERE clause of the user query.
C. Use Athena partition projection based on the S3 bucket prefix.
D. Transform the data that is in the S3 bucket to Apache Parquet format.
E. Use the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects.

Correct Answer:

A. Create an AWS Glue partition index. Enable partition filtering.

B. Bucket the data based on a column that the data have in common in a WHERE clause of the user query.

C. Use Athena partition projection based on the S3 bucket prefix.

D. Transform the data that is in the S3 bucket to Apache Parquet format.

E. Use the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects.

Explanation:

Certainly! Let's break down why options A and C are the correct solutions to address the performance bottleneck caused by a large number of partitions in Amazon Athena queries.

Option A: Create an AWS Glue partition index. Enable partition filtering.

1. Partition Indexing: - AWS Glue partition indexing helps manage and expedite the retrieval of partition metadata, which can significantly improve query planning times. By indexing partitions, Athena can quickly locate the relevant partitions needed for a query without scanning the entire partition metadata. - This is particularly useful when dealing with large datasets that have numerous partitions, as it reduces the overhead associated with querying partition metadata.

2. Partition Filtering: - Partition filtering ensures that only the necessary partitions are scanned based on the query's predicates (conditions specified in the WHERE clause). This means that Athena does not need to check every partition, which speeds up query execution and reduces data scanning costs. - Enabling partition filtering works well with partition indexing to optimize query performance.

Option C: Use Athena partition projection based on the S3 bucket prefix.

1. Partition Projection: - Partition projection allows you to define partitioning schemes for your datasets without physically storing metadata for each partition in the AWS Glue Data Catalog. Instead, partitions are inferred based on naming conventions, such as prefixes in S3. - This reduces the load on the Data Catalog and speeds up query planning because Athena can directly compute the partitions involved in a query without fetching metadata for every partition.

2. Efficiency with Large Datasets: - For datasets with a large number of partitions, partition projection can significantly reduce the query planning time because it eliminates the need for frequent catalog lookups. The partitions are computed dynamically, which is particularly efficient for queries that target a subset of partitions.

Why Not the Other Options?

- Option B (Bucketing): While bucketing can help with certain types of query optimization, it primarily affects data organization and storage, not partitioning. It doesn't directly address the issue of query planning time related to partition metadata.

- Option D (Transform to Parquet): Converting data to Apache Parquet format can improve query performance due to Parquet's columnar storage benefits, but it does not directly resolve the issue of a large number of partitions affecting query planning time.

- Option E (Combining Smaller Objects): Using the S3DistCP utility to combine smaller objects can help with reducing the number of S3 API calls and improving read performance, but it does not address partition metadata issues in Athena queries.

In summary, options A and C are specifically designed to address the challenge of managing and optimizing partition metadata handling, which directly targets the query planning bottleneck faced by the data engineer.

Question # 30

Date: February 02, 2024

A data engineering team is using an Amazon Redshift data warehouse for operational reporting. The team wants to prevent performance issues that might result from long- running queries. A data engineer must choose a system table in Amazon Redshift to record anomalies when a query optimizer identifies conditions that might indicate performance issues.

Which table views should the data engineer use to meet this requirement?

A. STL_USAGE_CONTROL
B. STL_ALERT_EVENT_LOG
C. STL_QUERY_METRICS
D. STL_PLAN_INFO

Correct Answer:

A. STL_USAGE_CONTROL

B. STL_ALERT_EVENT_LOG

C. STL_QUERY_METRICS

D. STL_PLAN_INFO

Explanation:

In Amazon Redshift, monitoring and optimizing query performance is crucial, especially for a data engineering team that relies on a data warehouse for operational reporting. When queries run longer than expected, it could indicate potential performance issues, which can be a result of various underlying factors such as inefficient query design, suboptimal resource allocation, or system constraints.

The correct answer to the question is B: STL_ALERT_EVENT_LOG. Let's break down why this is the appropriate choice:

1. STL_ALERT_EVENT_LOG: This system table is specifically designed to log events related to performance issues that the query optimizer detects in Amazon Redshift. The query optimizer in Redshift evaluates the execution plans of queries and identifies any anomalies or conditions that could lead to inefficient query execution. These alerts might include information about missing statistics, suboptimal join conditions, or other factors that could degrade performance. By using this table, the data engineering team can capture and analyze these alerts to proactively address potential performance problems before they escalate.

2. Why Not the Other Options? - STL_USAGE_CONTROL: This table logs information about usage-based limits and controls, such as concurrency scaling and usage limits set on the cluster. It is not focused on query performance anomalies. - STL_QUERY_METRICS: While this table provides detailed metrics for query execution, including time taken and resources used, it does not specifically log alerts or anomalies identified by the query optimizer. - STL_PLAN_INFO: This table contains information about the execution plan of queries but does not provide alerts about performance anomalies. It is more about the structure of the query execution rather than identifying potential performance issues.

In summary, the STL_ALERT_EVENT_LOG table is tailored for capturing alerts related to performance anomalies detected by the query optimizer. It is a critical tool for the data engineering team to monitor, diagnose, and optimize query performance in Amazon Redshift, making it the correct choice for the requirement stated in the question.

Question # 31

Date: February 03, 2024

A company is building an analytics solution. The solution uses Amazon S3 for data lake storage and Amazon Redshift for a data warehouse. The company wants to use Amazon Redshift Spectrum to query the data that is in Amazon S3.

Which actions will provide the FASTEST queries? (Choose two.)

A. Use gzip compression to compress individual files to sizes that are between 1 GB and 5 GB.
B. Use a columnar storage file format.
C. Partition the data based on the most common query predicates.
D. Split the data into files that are less than 10 KB.
E. Use file formats that are not splittable.

Correct Answer:

A. Use gzip compression to compress individual files to sizes that are between 1 GB and 5 GB.

B. Use a columnar storage file format.

C. Partition the data based on the most common query predicates.

D. Split the data into files that are less than 10 KB.

E. Use file formats that are not splittable.

Explanation:

To understand why the correct answer is B and C, let's break down the advantages of each option in the context of optimizing query performance for Amazon Redshift Spectrum, which allows querying data stored in Amazon S3 directly from Amazon Redshift.

B. Use a columnar storage file format.

Using a columnar storage file format, such as Apache Parquet or Apache ORC, is advantageous for several reasons:

1. Efficient Data Retrieval: Columnar formats store data by columns rather than rows. When a query only requires certain columns, Redshift Spectrum only needs to read the relevant data, significantly reducing the amount of data scanned compared to row-based formats. This leads to faster query performance since less data is being read and processed.

2. Compression: Columnar formats typically support better compression, which reduces the amount of data that needs to be read from S3. Less data transfer from S3 to the processing layer results in faster query times.

3. Optimized for Analytics: These formats are specifically designed for complex analytical queries, making them a natural fit for Redshift Spectrum's use case.

C. Partition the data based on the most common query predicates.

Partitioning data strategically is crucial for optimizing query performance:

1. Reduced Data Scanning: By partitioning data based on common query predicates (e.g., date, region, etc.), queries can skip entire partitions that do not match the query criteria. This reduces the volume of data that needs to be scanned, leading to faster queries.

2. Improved Query Efficiency: When data is partitioned effectively, Redshift Spectrum can quickly locate the relevant data segments, which enhances query performance. This is particularly beneficial when dealing with large datasets, as it limits the scope of data processing to only what's necessary for the query.

Now, let's briefly address the other options to clarify why they are not as effective for the fastest query performance:

A. Use gzip compression to compress individual files to sizes that are between 1 GB and 5 GB.

While gzip compression can reduce file sizes, it is not designed for efficient columnar data access, and larger file sizes can lead to slower query performance due to the increased time to decompress and process these files. Moreover, file sizes between 1 GB and 5 GB might still be too large for optimal query performance, as Redshift Spectrum works best with smaller and more manageable file sizes in columnar formats.

D. Split the data into files that are less than 10 KB.

Files that are too small can lead to inefficient query performance because each file incurs a certain amount of overhead for metadata and processing. Very small files can overwhelm the system with too many requests, leading to increased latency and slower query execution.

E. Use file formats that are not splittable.

Non-splittable file formats (like certain compressed formats) mean that the entire file must be read even if only part of it is needed, leading to inefficient data processing and slower query performance. This is the opposite of what is desired for fast queries, where you want to minimize the amount of data read.

In summary, using a columnar storage file format and partitioning data based on query predicates are the most effective strategies for optimizing query performance with Amazon Redshift Spectrum.

Question # 32

Date: February 03, 2024

A company uses Amazon RDS to store transactional data. The company runs an RDS DB instance in a private subnet. A developer wrote an AWS Lambda function with default settings to insert, update, or delete data in the DB instance.

The developer needs to give the Lambda function the ability to connect to the DB instance privately without using the public internet.

Which combination of steps will meet this requirement with the LEAST operational overhead? (Choose two.)

A. Turn on the public access setting for the DB instance.
B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.
C. Configure the Lambda function to run in the same subnet that the DB instance uses.
D. Attach the same security group to the Lambda function and the DB instance. Include a self-referencing rule that allows access through the database port.
E. Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port.

Correct Answer:

A. Turn on the public access setting for the DB instance.

B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.

C. Configure the Lambda function to run in the same subnet that the DB instance uses.

D. Attach the same security group to the Lambda function and the DB instance. Include a self-referencing rule that allows access through the database port.

E. Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port.

Explanation:

When dealing with Amazon RDS and AWS Lambda, especially in a private subnet, the primary concern is ensuring secure and private connections without exposing resources to the public internet. The goal is to enable Lambda to access the RDS instance efficiently while minimizing operational complexity. Let's break down why options C and D are the correct choices:

Option C: Configure the Lambda function to run in the same subnet that the DB instance uses.

Running the Lambda function in the same subnet as the RDS instance is crucial for maintaining network efficiency and security. By placing both resources in the same subnet, the Lambda function can access the RDS instance directly without needing to traverse the public internet or other subnets, which can introduce latency and security risks. This setup ensures that traffic between the Lambda function and the RDS instance remains within the same private network, providing a direct and secure path for data operations.

Option D: Attach the same security group to the Lambda function and the DB instance. Include a self-referencing rule that allows access through the database port.

Security groups act as virtual firewalls that control inbound and outbound traffic to AWS resources. By attaching the same security group to both the Lambda function and the RDS instance, you simplify network management and ensure that both resources can communicate with each other securely. Adding a self-referencing rule in the security group allows traffic on the database port (typically port 3306 for MySQL, for example) between the resources that share the security group. This rule ensures that the Lambda function can interact with the RDS instance over the necessary port without additional configuration or exposure to the public network.

Why the other options are not correct:

- Option A: Turning on public access for the DB instance would expose it to the internet, which is unnecessary and risky when the requirement is to connect privately. This increases the attack surface and goes against best practices for security.

- Option B: While updating the security group to allow Lambda function invocations on the database port seems logical, it doesn't address the setup in terms of network configurations and routing, especially given the default settings of the Lambda function.

- Option E: Network ACLs (Access Control Lists) manage traffic at the subnet level and are not specific to individual resources. While they can be used for additional security, they add complexity and are not necessary for the specific requirement to allow Lambda to access RDS privately when security groups are already in use.

By combining options C and D, you ensure that the Lambda function can communicate securely and privately with the RDS instance with minimal setup and operational overhead.

Question # 33

Date: February 01, 2024

A company has a frontend ReactJS website that uses Amazon API Gateway to invoke REST APIs. The APIs perform the functionality of the website. A data engineer needs to write a Python script that can be occasionally invoked through API Gateway. The code must return results to API Gateway.

Which solution will meet these requirements with the LEAST operational overhead?

A. Deploy a custom Python script on an Amazon Elastic Container Service (Amazon ECS) cluster.
B. Create an AWS Lambda Python function with provisioned concurrency.
C. Deploy a custom Python script that can integrate with API Gateway on Amazon Elastic Kubernetes Service (Amazon EKS).
D. Create an AWS Lambda function. Ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events.

Correct Answer:

A. Deploy a custom Python script on an Amazon Elastic Container Service (Amazon ECS) cluster.

B. Create an AWS Lambda Python function with provisioned concurrency.

C. Deploy a custom Python script that can integrate with API Gateway on Amazon Elastic Kubernetes Service (Amazon EKS).

D. Create an AWS Lambda function. Ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events.

Explanation:

The correct answer is B: Create an AWS Lambda Python function with provisioned concurrency.

Here's why this is the best solution with the least operational overhead:

1. AWS Lambda Overview: AWS Lambda is a serverless compute service that allows you to run code without having to provision or manage servers. You simply upload your code, and Lambda takes care of everything required to run and scale your code with high availability.

2. Operational Overhead: Using AWS Lambda significantly reduces operational overhead because there is no need to manage infrastructure. You don't have to worry about server maintenance, scaling, patching, or monitoring the underlying hardware, which you would have to do with Amazon ECS or Amazon EKS.

3. Provisioned Concurrency: Provisioned concurrency is a feature of AWS Lambda that keeps functions initialized and ready to respond in double-digit milliseconds. This ensures that the function is always warm and can handle requests immediately, which is ideal for applications that cannot afford latency spikes caused by cold starts.

4. Integration with API Gateway: Lambda functions integrate seamlessly with Amazon API Gateway, allowing you to invoke the function via HTTP requests. This makes it straightforward to set up the required connectivity between the frontend ReactJS website and the backend logic implemented in the Lambda function.

5. Cost-Effectiveness: Lambda's pay-per-use pricing model means you only pay for the compute time you consume. With provisioned concurrency, you pay for the amount of concurrency you configure and the duration your code runs, which can be more cost-effective than running containers on ECS or EKS, especially for occasionally invoked scripts.

6. Simplicity: Creating a Lambda function with provisioned concurrency is simpler than deploying and maintaining a containerized solution on ECS or EKS. With Lambda, you focus on writing and deploying your code, while AWS handles the infrastructure.

In contrast, options A and C involve deploying and managing containerized applications on ECS and EKS, which introduce additional complexity and operational overhead compared to the serverless approach of Lambda. Option D involves keeping the Lambda function warm through scheduled invocations, which is a workaround that adds unnecessary complexity compared to using provisioned concurrency directly. Therefore, option B provides the simplest and most efficient solution for the given requirements.

Question # 34

Date: February 06, 2024

A company has a production AWS account that runs company workloads. The company's security team created a security AWS account to store and analyze security logs from the production AWS account. The security logs in the production AWS account are stored in Amazon CloudWatch Logs.

The company needs to use Amazon Kinesis Data Streams to deliver the security logs to the security AWS account.

Which solution will meet these requirements?

A. Create a destination data stream in the production AWS account. In the security AWS account, create an IAM role that has cross-account permissions to Kinesis Data Streams in the production AWS account.
B. Create a destination data stream in the security AWS account. Create an IAM role and a trust policy to grant CloudWatch Logs the permission to put data into the stream. Create a subscription filter in the security AWS account.
C. Create a destination data stream in the production AWS account. In the production AWS account, create an IAM role that has cross-account permissions to Kinesis Data Streams in the security AWS account.
D. Create a destination data stream in the security AWS account. Create an IAM role and a trust policy to grant CloudWatch Logs the permission to put data into the stream. Create a subscription filter in the production AWS account.

Correct Answer:

A. Create a destination data stream in the production AWS account. In the security AWS account, create an IAM role that has cross-account permissions to Kinesis Data Streams in the production AWS account.

B. Create a destination data stream in the security AWS account. Create an IAM role and a trust policy to grant CloudWatch Logs the permission to put data into the stream. Create a subscription filter in the security AWS account.

C. Create a destination data stream in the production AWS account. In the production AWS account, create an IAM role that has cross-account permissions to Kinesis Data Streams in the security AWS account.

D. Create a destination data stream in the security AWS account. Create an IAM role and a trust policy to grant CloudWatch Logs the permission to put data into the stream. Create a subscription filter in the production AWS account.

Explanation:

Certainly! Let's break down why option D is the correct solution for the scenario described:

1. Objective: The company wants to transfer security logs from the production AWS account, where they are stored in Amazon CloudWatch Logs, to a security AWS account using Amazon Kinesis Data Streams.

2. Destination Data Stream: The requirement is to deliver logs to the security AWS account. Therefore, the destination data stream should logically reside in the security AWS account. This allows the security account to own and manage the data stream, ensuring that it has control over the logs once they are transferred.

3. IAM Role and Trust Policy: - An IAM role with appropriate permissions is necessary to allow CloudWatch Logs in the production account to send data to Kinesis Data Streams in the security account. - The trust policy is crucial because it defines which accounts or services are allowed to assume the IAM role. In this case, you need to allow the CloudWatch Logs service from the production account to assume the role and put data into the Kinesis Data Stream in the security account.

4. Subscription Filter: - The subscription filter is created in the production AWS account. This is because the logs originate from the production account's CloudWatch Logs. The subscription filter specifies what log data should be sent to the Kinesis Data Stream. It acts as a bridge, filtering the relevant log events and sending them to the specified Kinesis Data Stream in the security account.

5. Cross-Account Setup: - This setup involves configuring cross-account access, where the production account is configured to send data to the security account's Kinesis Data Stream. This is achieved through the IAM role and trust policy, ensuring that permissions are securely managed and logs are correctly transferred.

By setting up the destination data stream in the security AWS account and configuring the IAM role and trust policies appropriately, the company can securely and efficiently transfer security logs from the production account to the security account. This approach ensures that the security account has control over the log data once it's delivered, aligning with the company's need to analyze and store security logs separately from production workloads.

Question # 35

Date:

A company uses Amazon S3 to store semi-structured data in a transactional data lake. Some of the data files are small, but other data files are tens of terabytes.

A data engineer must perform a change data capture (CDC) operation to identify changed data from the data source. The data source sends a full snapshot as a JSON file every day and ingests the changed data into the data lake.

Which solution will capture the changed data MOST cost-effectively?

A. Create an AWS Lambda function to identify the changes between the previous data and the current data. Configure the Lambda function to ingest the changes into the data lake.
B. Ingest the data into Amazon RDS for MySQL. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.
C. Use an open source data lake format to merge the data source with the S3 data lake to insert the new data and update the existing data.
D. Ingest the data into an Amazon Aurora MySQL DB instance that runs Aurora Serverless. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

Correct Answer:

A. Create an AWS Lambda function to identify the changes between the previous data and the current data. Configure the Lambda function to ingest the changes into the data lake.

B. Ingest the data into Amazon RDS for MySQL. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

C. Use an open source data lake format to merge the data source with the S3 data lake to insert the new data and update the existing data.

D. Ingest the data into an Amazon Aurora MySQL DB instance that runs Aurora Serverless. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

Explanation:

The correct answer to this question is option C, which suggests using an open-source data lake format to merge the data source with the S3 data lake to insert the new data and update the existing data. Let's break down why this is the most cost-effective solution for the scenario described:

1. Open-Source Data Lake Format: Open-source data lake formats, such as Apache Hudi, Delta Lake, or Apache Iceberg, are specifically designed to handle change data capture (CDC) operations efficiently in data lakes stored on platforms like Amazon S3. These formats provide capabilities for upserts (update and insert operations), which are crucial for handling CDC scenarios where you need to track and manage changes over time.

2. Cost-Effectiveness: Using an open-source format directly on Amazon S3 is cost-effective because it reduces the need for additional infrastructure. You don't have to provision databases or manage additional services. The data is processed in place, leveraging the scalability and durability of S3 without incurring the costs associated with running database instances or serverless functions continuously.

3. Efficiency and Scalability: These data lake formats are designed for high performance and scalability, which is essential when dealing with large datasets that can be tens of terabytes in size. They optimize storage and manage metadata efficiently, ensuring that operations such as merging or querying datasets are performed quickly and without unnecessary overhead.

4. Simplified Data Management: By using an open-source data lake format, the data engineer can avoid complex data transfer operations and transformations that might be required when using other services like databases. These formats often come with built-in support for transaction management, schema evolution, and efficient file-level operations, which makes handling large files and frequent updates more straightforward.

5. Avoiding Unnecessary Database Use: Options B and D involve using relational databases (Amazon RDS for MySQL and Amazon Aurora Serverless), which can introduce additional costs and complexities, especially when dealing with very large datasets. These databases would have to be scaled to handle potentially massive daily data loads, which can be cost-prohibitive and might not be necessary if the data is mainly used for storage and analytical purposes.

6. Lambda Function Limitations: Option A suggests using AWS Lambda, which has execution time and memory limits. While Lambda can be useful for smaller-scale data processing tasks, it may not be the best fit for processing large datasets due to these constraints. Additionally, Lambda pricing is based on execution time and resource consumption, which might become costly for processing tens of terabytes of data daily.

Overall, option C provides a solution that is both cost-effective and technically sound for the scenario of capturing and managing changed data in an Amazon S3 data lake, especially when dealing with large volumes of semi-structured data.

Question # 36

Date: February 02, 2024

An airline company is collecting metrics about flight activities for analytics. The company is conducting a proof of concept (POC) test to show how analytics can provide insights that the company can use to increase on-time departures.

The POC test uses objects in Amazon S3 that contain the metrics in .csv format. The POC test uses Amazon Athena to query the data. The data is partitioned in the S3 bucket by date.

As the amount of data increases, the company wants to optimize the storage solution to improve query performance.

Which combination of solutions will meet these requirements? (Choose two.)

A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.
B. Use an S3 bucket that is in the same account that uses Athena to query the data.
C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.
D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.
E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Correct Answer:

A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.

B. Use an S3 bucket that is in the same account that uses Athena to query the data.

C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.

D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.

E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Explanation:

The correct answer is CE, which involves using an S3 bucket in the same AWS Region as Athena and preprocessing data to Apache Parquet format. Here's why these solutions are effective:

C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.

When you use an S3 bucket in the same AWS Region as Athena, you minimize data transfer latency. Data transfer between S3 and Athena is faster within the same region, which directly improves query performance. Reducing latency is crucial when dealing with large datasets, as even small delays can accumulate into significant wait times during query execution. Additionally, keeping your storage and compute resources in the same region can help reduce costs associated with data transfer fees, as inter-region data transfers typically incur additional charges.

E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Apache Parquet is a columnar storage file format that is highly efficient for analytical queries. Compared to CSV, Parquet is optimized for performance in analytics because it stores data in a column-wise fashion. This allows Athena to read only the relevant columns needed for a query, rather than scanning entire rows, which is the case with CSV files. This columnar storage format significantly reduces the amount of data that needs to be read during a query, leading to faster query performance and reduced data retrieval costs. Moreover, Parquet supports efficient compression and encoding schemes, further reducing the size of the data and enhancing performance. Preprocessing the data to Parquet before querying means that the data is already optimized for Athena's query patterns, resulting in quicker response times and more efficient use of resources.

Together, these choices optimize both the physical location of the data and the format in which it is stored, addressing key factors that affect query performance: data transfer speed and data processing efficiency.

Question # 37

Date: February 03, 2024

A data engineer must manage the ingestion of real-time streaming data into AWS. The data engineer wants to perform real-time analytics on the incoming streaming data by using time-based aggregations over a window of up to 30 minutes. The data engineer needs a solution that is highly fault tolerant.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use an AWS Lambda function that includes both the business and the analytics logic to perform time-based aggregations over a window of up to 30 minutes for the data in Amazon Kinesis Data Streams.
B. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data that might occasionally contain duplicates by using multiple types of aggregations.
C. Use an AWS Lambda function that includes both the business and the analytics logic to perform aggregations for a tumbling window of up to 30 minutes, based on the event timestamp.
D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data by using multiple types of aggregations to perform time-based analytics over a window of up to 30 minutes.

Correct Answer:

A. Use an AWS Lambda function that includes both the business and the analytics logic to perform time-based aggregations over a window of up to 30 minutes for the data in Amazon Kinesis Data Streams.

B. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data that might occasionally contain duplicates by using multiple types of aggregations.

C. Use an AWS Lambda function that includes both the business and the analytics logic to perform aggregations for a tumbling window of up to 30 minutes, based on the event timestamp.

D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data by using multiple types of aggregations to perform time-based analytics over a window of up to 30 minutes.

Explanation:

The correct answer is D: Use Amazon Managed Service for Apache Flink to analyze the data by using multiple types of aggregations to perform time-based analytics over a window of up to 30 minutes.

Let's break down why this is the best solution given the requirements:

1. Real-time Streaming Data Ingestion and Analytics: The scenario requires handling real-time data streams and performing analytics on this data as it arrives. Amazon Managed Service for Apache Flink is designed specifically for processing and analyzing streaming data in real-time. It allows you to execute complex event processing and real-time analytics on streaming data efficiently.

2. Time-based Aggregations: The requirement specifies performing time-based aggregations over a window of up to 30 minutes. Apache Flink is well-suited for this as it provides robust support for windowing operations, including tumbling, sliding, and session windows, which can handle time-based aggregations seamlessly.

3. Fault Tolerance: Apache Flink is inherently fault-tolerant. It supports stateful stream processing, and in the event of a failure, it can restore the state from a checkpoint, ensuring that the application can continue processing from where it left off without data loss. This makes it an excellent choice for applications that need high reliability.

4. Least Operational Overhead: The requirement emphasizes minimizing operational overhead. Amazon Managed Service for Apache Flink is a fully managed service, which means AWS handles much of the undifferentiated heavy lifting such as provisioning infrastructure, managing clusters, and scaling the application. This significantly reduces the operational burden on the data engineer, allowing them to focus on building the application logic rather than managing the infrastructure.

Comparing with other options: - AWS Lambda (Options A and C): While AWS Lambda can process streaming data, it is generally used for event-driven architectures and simple processing. It is not optimized for complex, stateful stream processing or time-based windows over long durations like 30 minutes. Additionally, managing state manually in Lambda could increase operational overhead. - Handling Duplicates (Option B): Option B mentions handling duplicates, which can be a common issue in streaming data. Apache Flink can handle duplicates effectively using features like exactly-once processing semantics, which further supports the need for a robust and reliable analytics solution.

In summary, Amazon Managed Service for Apache Flink offers the capabilities needed for performing complex real-time analytics with time-based aggregations, while also providing fault tolerance and minimizing operational overhead, making it the most suitable choice for the given requirements.

Question # 38

Date: February 04, 2024

A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage.

Which solution will meet these requirements with the LEAST operational overhead?

A. Create snapshots of the gp2 volumes. Create new gp3 volumes from the snapshots. Attach the new gp3 volumes to the EC2 instances.
B. Create new gp3 volumes. Gradually transfer the data to the new gp3 volumes. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.
C. Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.
D. Use AWS DataSync to create new gp3 volumes. Transfer the data from the original gp2 volumes to the new gp3 volumes.

Correct Answer:

A. Create snapshots of the gp2 volumes. Create new gp3 volumes from the snapshots. Attach the new gp3 volumes to the EC2 instances.

B. Create new gp3 volumes. Gradually transfer the data to the new gp3 volumes. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.

C. Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.

D. Use AWS DataSync to create new gp3 volumes. Transfer the data from the original gp2 volumes to the new gp3 volumes.

Explanation:

The question is about upgrading Amazon Elastic Block Store (EBS) volumes from the General Purpose SSD gp2 type to the gp3 type with minimal operational overhead and without causing interruptions or data loss for Amazon EC2 instances.

Let's explore why option C is the correct answer:

Option C suggests simply changing the volume type of existing gp2 volumes to gp3. This is the most straightforward solution provided in the options, as it leverages a feature that AWS offers: the ability to modify the volume type of an existing EBS volume without detaching it. This modification can be done directly from the AWS Management Console, AWS CLI, or through the AWS SDKs.

Here’s why this option meets the requirements with the least operational overhead:

1. Simplicity: Changing the volume type does not require creating new volumes or transferring data manually. The modification process is handled by AWS, making it a seamless transition.

2. No Data Transfer Required: Since you are modifying the existing volume, there is no need to create snapshots, copy data, or manage any manual data transfer process. This reduces complexity and the potential for errors.

3. Minimal Downtime: Changing the volume type in place minimizes the risk of downtime. The operation can be performed while the instance is running, and AWS manages the migration of the underlying hardware resources.

4. Automatic Handling by AWS: AWS takes care of the necessary backend tasks to upgrade the volume type from gp2 to gp3, ensuring that the change does not interrupt the instance operations or cause data loss.

In contrast, the other options involve additional steps that introduce complexity or potential for interruptions:

- Option A involves creating snapshots and new volumes, which adds steps and risk of errors in the attachment process. - Option B requires manually transferring data, which is operationally intensive and error-prone. - Option D suggests using AWS DataSync, which is unnecessary for this task and adds complexity by introducing an additional service for data transfer.

Overall, option C provides a direct, efficient, and low-overhead solution to upgrade EBS volumes from gp2 to gp3, aligning perfectly with the company's requirements to minimize operational complexity and avoid interruptions.

Question # 39

Date: February 03, 2024

A company is migrating its database servers from Amazon EC2 instances that run Microsoft SQL Server to Amazon RDS for Microsoft SQL Server DB instances. The company's analytics team must export large data elements every day until the migration is complete. The data elements are the result of SQL joins across multiple tables. The data must be in Apache Parquet format. The analytics team must store the data in Amazon S3.

Which solution will meet these requirements in the MOST operationally efficient way?

A. Create a view in the EC2 instance-based SQL Server databases that contains the required data elements. Create an AWS Glue job that selects the data directly from the view and transfers the data in Parquet format to an S3 bucket. Schedule the AWS Glue job to run every day.
B. Schedule SQL Server Agent to run a daily SQL query that selects the desired data elements from the EC2 instance-based SQL Server databases. Configure the query to direct the output .csv objects to an S3 bucket. Create an S3 event that invokes an AWS Lambda function to transform the output format from .csv to Parquet.
C. Use a SQL query to create a view in the EC2 instance-based SQL Server databases that contains the required data elements. Create and run an AWS Glue crawler to read the view. Create an AWS Glue job that retrieves the data and transfers the data in Parquet format to an S3 bucket. Schedule the AWS Glue job to run every day.
D. Create an AWS Lambda function that queries the EC2 instance-based databases by using Java Database Connectivity (JDBC). Configure the Lambda function to retrieve the required data, transform the data into Parquet format, and transfer the data into an S3 bucket. Use Amazon EventBridge to schedule the Lambda function to run every day.

Correct Answer:

A. Create a view in the EC2 instance-based SQL Server databases that contains the required data elements. Create an AWS Glue job that selects the data directly from the view and transfers the data in Parquet format to an S3 bucket. Schedule the AWS Glue job to run every day.

B. Schedule SQL Server Agent to run a daily SQL query that selects the desired data elements from the EC2 instance-based SQL Server databases. Configure the query to direct the output .csv objects to an S3 bucket. Create an S3 event that invokes an AWS Lambda function to transform the output format from .csv to Parquet.

C. Use a SQL query to create a view in the EC2 instance-based SQL Server databases that contains the required data elements. Create and run an AWS Glue crawler to read the view. Create an AWS Glue job that retrieves the data and transfers the data in Parquet format to an S3 bucket. Schedule the AWS Glue job to run every day.

D. Create an AWS Lambda function that queries the EC2 instance-based databases by using Java Database Connectivity (JDBC). Configure the Lambda function to retrieve the required data, transform the data into Parquet format, and transfer the data into an S3 bucket. Use Amazon EventBridge to schedule the Lambda function to run every day.

Explanation:

The correct answer is C. Let's break down why this solution is the most operationally efficient for the given scenario:

1. Use of AWS Glue Crawler and Job: - AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics. It is well-suited for this task because it can automate the discovery of data and store the associated metadata (i.e., table definitions, schema) in the AWS Glue Data Catalog. - The Glue Crawler can be used to read the data from the SQL Server view, automatically detecting the schema and creating metadata, which is essential for subsequent data processing tasks.

2. Creating a View: - By creating a view in the SQL Server database, the company can simplify the extraction process, as the view will consolidate and encapsulate the logic of the SQL joins across multiple tables. This makes it easier to query the necessary data elements without having to rewrite complex SQL queries each time.

3. Transformation to Parquet Format: - AWS Glue jobs are designed to transform data from its source format into a target format. In this scenario, the job will read the data from the SQL Server view and convert it into Apache Parquet format. Parquet is a columnar storage file format that is highly efficient for both storage and query performance, especially for large datasets.

4. Operational Efficiency: - This solution leverages AWS Glue's serverless architecture, which means there is no need to manage infrastructure, making it highly scalable and cost-effective. AWS Glue automates much of the heavy lifting involved in data transformation and loading processes. - By scheduling the Glue job to run every day, the company ensures that the data in S3 is consistently updated without manual intervention, enhancing operational efficiency.

5. Data Storage in Amazon S3: - S3 is a highly durable and scalable object storage service that is ideal for storing large volumes of data. By storing the transformed data in S3, the company can ensure that the analytics team has access to the data in a cost-effective and reliable manner.

In summary, option C is the most operationally efficient solution because it utilizes AWS Glue to automate the data extraction, transformation, and loading (ETL) process, minimizes manual intervention, and effectively manages and handles large datasets while converting them into a format suitable for analytics.

Question # 40

Date: March 07, 2024

A company uses Amazon RDS for MySQL as the database for a critical application. The database workload is mostly writes, with a small number of reads.

A data engineer notices that the CPU utilization of the DB instance is very high. The high CPU utilization is slowing down the application. The data engineer must reduce the CPU utilization of the DB Instance.

Which actions should the data engineer take to meet this requirement? (Choose two.)

A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization. Optimize the problematic queries.
B. Modify the database schema to include additional tables and indexes.

C. Reboot the RDS DB instance once each week.
D. Upgrade to a larger instance size.
E. Implement caching to reduce the database query load.

Correct Answer:

A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization. Optimize the problematic queries.

B. Modify the database schema to include additional tables and indexes.

C. Reboot the RDS DB instance once each week.

D. Upgrade to a larger instance size.

E. Implement caching to reduce the database query load.

Explanation:

The correct answers to this question are A and D, and here's why:

A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization. Optimize the problematic queries.

Performance Insights is a powerful tool provided by Amazon RDS that helps you monitor and analyze the performance of your database. It allows you to identify which SQL queries are consuming the most resources, such as CPU. In this scenario, since the database workload is mostly writes and the CPU utilization is very high, it is likely that certain queries are not optimized, leading to increased CPU load. By using Performance Insights, the data engineer can pinpoint these problematic queries and optimize them, which could significantly reduce the CPU utilization. Optimizing queries may involve rewriting inefficient SQL, using better indexing strategies, or restructuring the queries to be more efficient.

D. Upgrade to a larger instance size.

Upgrading to a larger instance size is a straightforward approach to handle high CPU utilization. When the CPU on the current instance is consistently high, it indicates that the instance does not have enough processing power to handle the workload efficiently. By choosing a larger instance size, the company can benefit from more CPU resources, which can help in accommodating the current demand without slowing down the application. This is a direct way to immediately increase the available computational capacity, which can alleviate the high CPU load and improve performance.

These two actions address both the root cause by optimizing the workload (A) and the symptoms by increasing capacity (D). This combined approach helps ensure that the database can handle the current and potentially future workloads more effectively.

Question # 41

Date: January 28, 2024

A data engineer must ingest a source of structured data that is in .csv format into an Amazon S3 data lake. The .csv files contain 15 columns. Data analysts need to run Amazon Athena queries on one or two columns of the dataset. The data analysts rarely query the entire file.

Which solution will meet these requirements MOST cost-effectively?

A. Use an AWS Glue PySpark job to ingest the source data into the data lake in .csv format.
B. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to ingest the data into the data lake in JSON format.
C. Use an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format.
D. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to write the data into the data lake in Apache Parquet format.

Correct Answer:

A. Use an AWS Glue PySpark job to ingest the source data into the data lake in .csv format.

B. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to ingest the data into the data lake in JSON format.

C. Use an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format.

D. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to write the data into the data lake in Apache Parquet format.

Explanation:

The correct answer here is D: Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to write the data into the data lake in Apache Parquet format.

Here's why this solution is the most cost-effective:

1. Columnar Storage Format: Apache Parquet is a columnar storage file format, which means it stores data by column rather than by row. This is highly efficient for querying because it allows Amazon Athena to read only the columns needed for the query, instead of scanning the entire dataset. Since the data analysts are primarily querying one or two columns, using a columnar format like Parquet can significantly reduce the amount of data that needs to be read, which in turn reduces query costs.

2. Compression and Performance: Parquet also offers better compression and performance benefits compared to row-based formats like CSV or JSON. It compresses data effectively, which not only saves storage costs in Amazon S3 but also reduces the amount of data transferred during queries, leading to lower query execution costs.

3. Optimized for Analytics: Parquet is specifically optimized for analytical queries, making it a natural fit for use with Amazon Athena. Athena's cost model is based on the amount of data scanned, so using Parquet, which minimizes the data scanned, directly contributes to more cost-efficient queries.

4. Scalability: As the dataset grows, the benefits of using a columnar format become even more pronounced. The storage and query cost savings scale with the size of the dataset, making Parquet a sustainable choice for long-term data lake solutions.

In contrast, options A, B, and C do not offer the same level of cost efficiency for the given scenario: - Option A continues to store data in CSV format, which would increase query costs because Athena would need to scan the entire dataset even if only a few columns are queried. - Option B suggests using JSON format, which, like CSV, is not a columnar format and would result in higher data scanning costs. - Option C mentions using Apache Avro, which is a good format for row-based data and supports schema evolution but is not as efficient as Parquet for analytical queries that frequently access only subsets of columns.

Overall, Apache Parquet provides a cost-effective and performance-optimized solution for querying data in Amazon Athena, especially when the database is queried on specific columns frequently.

Question # 42

Date: January 28, 2024

A company has five offices in different AWS Regions. Each office has its own human resources (HR) department that uses a unique IAM role. The company stores employee records in a data lake that is based on Amazon S3 storage.

A data engineering team needs to limit access to the records. Each HR department should be able to access records for only employees who are within the HR department's Region.

Which combination of steps should the data engineering team take to meet this requirement with the LEAST operational overhead? (Choose two.)

A. Use data filters for each Region to register the S3 paths as data locations.
B. Register the S3 path as an AWS Lake Formation location.
C. Modify the IAM roles of the HR departments to add a data filter for each department's Region.
D. Enable fine-grained access control in AWS Lake Formation. Add a data filter for each Region.
E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region.

Correct Answer:

A. Use data filters for each Region to register the S3 paths as data locations.

B. Register the S3 path as an AWS Lake Formation location.

C. Modify the IAM roles of the HR departments to add a data filter for each department's Region.

D. Enable fine-grained access control in AWS Lake Formation. Add a data filter for each Region.

E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region.

Explanation:

To address the requirement of limiting access to employee records by region with minimal operational overhead, let's break down the selected options, B and D, and understand why they are the best choices:

B. Register the S3 path as an AWS Lake Formation location.

- AWS Lake Formation: It's a service that simplifies the process of setting up a secure data lake. It provides capabilities for data ingestion, storage, cataloging, and securing access. - Registering the S3 path: By registering the S3 bucket (where the employee records are stored) as a location in Lake Formation, you can manage permissions and access controls in a centralized manner. This is crucial because it allows for easier management of data access policies without having to manually configure permissions directly on each S3 bucket or object.

D. Enable fine-grained access control in AWS Lake Formation. Add a data filter for each Region.

- Fine-grained access control: Lake Formation allows you to define permissions not only at the table or database level but also at a more detailed level like specific columns or rows. This capability is essential for controlling access based on specific criteria, such as the employee's region. - Data filters for each Region: By utilizing data filters, the data engineering team can specify which records (e.g., based on region) each HR department can access. This ensures that the HR department in a specific region can only see employee records relevant to their region. This approach minimizes the need for complex IAM roles and policies, reducing operational overhead.

The other options are less optimal due to the following reasons:

- A. Use data filters for each Region to register the S3 paths as data locations. This would involve manually managing multiple data locations, which increases complexity and operational overhead. - C. Modify the IAM roles of the HR departments to add a data filter for each department's Region. Managing multiple IAM roles with specific data filters can become cumbersome and error-prone, especially as the number of regions and departments grows.

- E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region. This approach requires maintaining multiple S3 buckets and individual IAM policies, which increases management complexity and operational overhead.

By leveraging Lake Formation's capabilities to manage and control access from a central point with minimal configuration changes, options B and D provide a streamlined solution that effectively meets the company's requirement with the least operational overhead.

Question # 43

Date: January 28, 2024

A company uses AWS Step Functions to orchestrate a data pipeline. The pipeline consists of Amazon EMR jobs that ingest data from data sources and store the data in an Amazon S3 bucket. The pipeline also includes EMR jobs that load the data to Amazon Redshift.

The company's cloud infrastructure team manually built a Step Functions state machine. The cloud infrastructure team launched an EMR cluster into a VPC to support the EMR jobs. However, the deployed Step Functions state machine is not able to run the EMR jobs.

Which combination of steps should the company take to identify the reason the Step Functions state machine is not able to run the EMR jobs? (Choose two.)

A. Use AWS CloudFormation to automate the Step Functions state machine deployment. Create a step to pause the state machine during the EMR jobs that fail. Configure the step to wait for a human user to send approval through an email message. Include details of the EMR task in the email message for further analysis.
B. Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. Verify that the Step Functions state machine code also includes IAM permissions to access the Amazon S3 buckets that the EMR jobs use. Use Access Analyzer for S3 to check the S3 access properties.
C. Check for entries in Amazon CloudWatch for the newly created EMR cluster. Change the AWS Step Functions state machine code to use Amazon EMR on EKS. Change the IAM access policies and the security group configuration for the Step Functions state machine code to reflect inclusion of Amazon Elastic Kubernetes Service (Amazon EKS).
D. Query the flow logs for the VPC. Determine whether the traffic that originates from the EMR cluster can successfully reach the data providers. Determine whether any security group that might be attached to the Amazon EMR cluster allows connections to the data source servers on the informed ports.
E. Check the retry scenarios that the company configured for the EMR jobs. Increase the number of seconds in the interval between each EMR task. Validate that each fallback state has the appropriate catch for each decision state. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

Correct Answer:

A. Use AWS CloudFormation to automate the Step Functions state machine deployment. Create a step to pause the state machine during the EMR jobs that fail. Configure the step to wait for a human user to send approval through an email message. Include details of the EMR task in the email message for further analysis.

B. Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. Verify that the Step Functions state machine code also includes IAM permissions to access the Amazon S3 buckets that the EMR jobs use. Use Access Analyzer for S3 to check the S3 access properties.

C. Check for entries in Amazon CloudWatch for the newly created EMR cluster. Change the AWS Step Functions state machine code to use Amazon EMR on EKS. Change the IAM access policies and the security group configuration for the Step Functions state machine code to reflect inclusion of Amazon Elastic Kubernetes Service (Amazon EKS).

D. Query the flow logs for the VPC. Determine whether the traffic that originates from the EMR cluster can successfully reach the data providers. Determine whether any security group that might be attached to the Amazon EMR cluster allows connections to the data source servers on the informed ports.

E. Check the retry scenarios that the company configured for the EMR jobs. Increase the number of seconds in the interval between each EMR task. Validate that each fallback state has the appropriate catch for each decision state. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

Explanation:

The question involves troubleshooting why an AWS Step Functions state machine is not able to run EMR jobs. The correct answer is option B and D, and here's why each of these steps is crucial:

Option B: Verify IAM Permissions - Importance of IAM Permissions: AWS Identity and Access Management (IAM) permissions are critical for ensuring that the Step Functions state machine can perform the necessary actions, such as creating and running EMR jobs and accessing Amazon S3 buckets. If the state machine does not have the appropriate permissions, it won't be able to execute the tasks as required. - Access Analyzer for S3: This tool helps in checking the S3 access properties, ensuring that the state machine has the necessary permissions to interact with S3 buckets. It is essential because the EMR jobs need to read from or write to S3, and without proper access, the jobs will fail. - Overall Impact: By verifying and correcting the IAM permissions, the state machine can properly execute EMR jobs and access necessary data, solving a common issue related to permission denials.

Option D: Query VPC Flow Logs - Understanding VPC Flow Logs: These logs provide visibility into the network traffic within a VPC, which can help identify if there are connectivity issues. If the traffic from the EMR cluster isn't reaching the data sources correctly due to network misconfigurations, the jobs will fail. - Security Group Configurations: Security groups act as virtual firewalls, controlling inbound and outbound traffic. By examining them, one can ensure that the EMR cluster can communicate with data sources over the right ports and protocols. - Network Troubleshooting: By analyzing the flow logs, the company can identify network or connectivity issues that could prevent the EMR cluster from functioning correctly within the pipeline.

Together, these steps encompass both permission and network configuration checks, which are the primary areas where issues typically arise in cloud infrastructure setups involving multiple AWS services. By addressing these two areas, the company can effectively troubleshoot and resolve why the Step Functions state machine is not executing the EMR jobs as expected.

Question # 44

Date: January 28, 2024

A company is developing an application that runs on Amazon EC2 instances. Currently, the data that the application generates is temporary. However, the company needs to persist the data, even if the EC2 instances are terminated.

A data engineer must launch new EC2 instances from an Amazon Machine Image (AMI) and configure the instances to preserve the data.

Which solution will meet this requirement?

A. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data. Apply the default settings to the EC2 instances.
B. Launch new EC2 instances by using an AMI that is backed by a root Amazon Elastic Block Store (Amazon EBS) volume that contains the application data. Apply the default settings to the EC2 instances.
C. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data. Apply the default settings to the EC2 instances.
D. Launch new EC2 instances by using an AMI that is backed by an Amazon Elastic Block Store (Amazon EBS) volume. Attach an additional EC2 instance store volume to contain the application data. Apply the default settings to the EC2 instances.

Correct Answer:

A. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data. Apply the default settings to the EC2 instances.

B. Launch new EC2 instances by using an AMI that is backed by a root Amazon Elastic Block Store (Amazon EBS) volume that contains the application data. Apply the default settings to the EC2 instances.

C. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data. Apply the default settings to the EC2 instances.

D. Launch new EC2 instances by using an AMI that is backed by an Amazon Elastic Block Store (Amazon EBS) volume. Attach an additional EC2 instance store volume to contain the application data. Apply the default settings to the EC2 instances.

Explanation:

The correct answer is C: Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data. Apply the default settings to the EC2 instances.

Here's why this solution meets the requirement:

1. Understanding the Storage Options: - EC2 Instance Store: This is temporary storage that's physically attached to the host computer. It's ephemeral, meaning that data is lost when the instance is stopped or terminated. It's fast but not durable for data persistence. - Amazon Elastic Block Store (EBS): This is a persistent block storage service. Data stored on EBS volumes is retained even after an EC2 instance is stopped or terminated, making it suitable for data that needs to persist beyond the lifetime of the instance.

2. Requirement for Data Persistence: The company needs to ensure that the data generated by the application persists even after the EC2 instances are terminated. This rules out relying solely on instance store volumes because they don't persist data.

3. Solution Explanation: - By launching EC2 instances using an AMI backed by an instance store, you can take advantage of the instance store's speed for temporary data needs. - Attaching an Amazon EBS volume ensures that any important data that needs to be retained is stored on a durable medium. This EBS volume can be detached and reattached to new instances or used independently, ensuring data persistence beyond the lifecycle of any single EC2 instance.

4. Why Other Options Are Not Suitable: - Option A: Relies entirely on instance store volumes, which do not persist data after instance termination, failing to meet the requirement for data persistence. - Option B: Uses an EBS-backed AMI but suggests applying default settings, which typically don't include additional EBS volumes specifically for persistent data. It doesn't explicitly ensure that the application data will be stored on a separate, persistent EBS volume. - Option D: Suggests attaching an additional instance store volume, which does not solve the problem of data persistence because instance store volumes are not durable.

Therefore, the combination of leveraging the instance store for its performance and attaching an EBS volume for data persistence makes Option C the most effective solution to meet the company's requirements.

Question # 45

Date: February 02, 2024

A company uses Amazon Athena to run SQL queries for extract, transform, and load (ETL) tasks by using Create Table As Select (CTAS). The company must use Apache Spark instead of SQL to generate analytics.

Which solution will give the company the ability to use Spark to access Athena?

A. Athena query settings
B. Athena workgroup
C. Athena data source
D. Athena query editor

Correct Answer:

A. Athena query settings

B. Athena workgroup

C. Athena data source

D. Athena query editor

Explanation:

Certainly! Let's break down why "Athena workgroup" is the correct solution for using Apache Spark to access Amazon Athena.

Amazon Athena is primarily a serverless, interactive query service that allows you to analyze data directly in Amazon S3 using standard SQL. However, if a company wants to use Apache Spark, which is a powerful open-source processing engine for big data, they need a way to integrate it with Athena.

Explanation:

1. Athena Workgroups: - Workgroups in Athena are a way to separate queries for different use cases, manage resources, and control query costs. They also help in managing query runtime and get better insights with query history and performance metrics. - Although Athena itself does not natively run Apache Spark queries, workgroups can be used as a part of a broader integration strategy where Athena is part of a data processing pipeline that includes Spark. - By setting up a workgroup, you can direct the queries and resources that are part of the ETL process managed by Spark, efficiently segregating them from other operations in Athena.

2. Integration with Apache Spark: - While Athena and Spark are fundamentally different (Athena uses Presto under the hood for SQL queries while Spark is its own processing engine), you can integrate them in a data pipeline. - Often, organizations use Spark for its superior processing capabilities on big datasets and then use Athena for querying the results stored in S3. - Workgroups can facilitate this by organizing the Athena side of the operations, ensuring that the Athena queries are correctly structured and managed, and that they work alongside Spark jobs.

3. Why Not the Other Options?: - A. Athena query settings: This relates to configuring specific query parameters but does not directly facilitate the use of Spark. - C. Athena data source: This refers to the source of data that Athena queries (like S3), but doesn't relate to using Spark. - D. Athena query editor: This is the interface used to write and execute SQL queries in Athena, which again doesn't help with integrating Spark.

In summary, while Athena does not run Spark itself, configuring an Athena workgroup is a step towards organizing and managing Athena's role in a larger data processing pipeline where Spark is used. This setup helps in efficiently managing the resources and workloads when using Spark alongside Athena for ETL tasks.

Question # 46

Date: January 29, 2024

A company needs to partition the Amazon S3 storage that the company uses for a data lake. The partitioning will use a path of the S3 object keys in the following format: s3://bucket/prefix/year=2023/month=01/day=01.

A data engineer must ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket.

Which solution will meet these requirements with the LEAST latency?

A. Schedule an AWS Glue crawler to run every morning.
B. Manually run the AWS Glue CreatePartition API twice each day.
C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create_partition API call.
D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Correct Answer:

A. Schedule an AWS Glue crawler to run every morning.

B. Manually run the AWS Glue CreatePartition API twice each day.

C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create_partition API call.

D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Explanation:

The question is about synchronizing new partitions in an Amazon S3 data lake with the AWS Glue Data Catalog with the least latency. Let's break down why option C is the correct choice:

1. Understanding Partitioning and Synchronization Needs: - Data lakes often use partitioned storage to organize and manage data efficiently. In this case, the company is using a partitioning strategy based on year, month, and day. - When new data is added to these partitions, the AWS Glue Data Catalog must be updated to reflect these changes so that queries can accurately access the latest data.

2. Evaluating the Options: - Option A (Schedule an AWS Glue crawler to run every morning): While AWS Glue crawlers can automatically discover partitions and update the Data Catalog, scheduling them to run at a specific time (like every morning) introduces latency. There is a delay between when the data is added to S3 and when the crawler runs to update the catalog. - Option B (Manually run the AWS Glue CreatePartition API twice each day): Manually running API calls is labor-intensive and prone to human error. It also introduces latency because updates to the catalog are not immediate and depend on the manual execution schedule. - Option C (Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create_partition API call): This option involves integrating the partition update directly into the data ingestion process. When new data is written to S3, the same process immediately triggers the AWS Glue `create_partition` API call using the Boto3 library. This approach ensures that the catalog is updated in real-time as soon as new data is added, providing the least latency. - Option D (Run the MSCK REPAIR TABLE command from the AWS Glue console): This command is useful for repairing partitions in Amazon Athena, which is compatible with Glue's Data Catalog. However, it is less automated and typically not the fastest method for updating the catalog compared to directly calling the API when new data is written.

3. Why Option C Has the Least Latency: - By embedding the partition creation logic directly into the data ingestion process, option C ensures that there is no delay between when new data is added and when the Glue Data Catalog is updated. This minimizes the time lag and ensures that new partitions are immediately available for querying. - Using Boto3, a popular AWS SDK for Python, allows for programmatic access to AWS services, making it highly efficient and reliable for managing AWS Glue tasks.

In summary, option C leverages automation and real-time processing to achieve minimal latency when synchronizing new partitions with the AWS Glue Data Catalog, making it the optimal choice for the requirement stated in the question.

Question # 47

Date: February 02, 2024

A media company uses software as a service (SaaS) applications to gather data by using third-party tools. The company needs to store the data in an Amazon S3 bucket. The company will use Amazon Redshift to perform analytics based on the data.

Which AWS service or feature will meet these requirements with the LEAST operational overhead?

A. Amazon Managed Streaming for Apache Kafka (Amazon MSK)
B. Amazon AppFlow
C. AWS Glue Data Catalog
D. Amazon Kinesis

Correct Answer:

A. Amazon Managed Streaming for Apache Kafka (Amazon MSK)

B. Amazon AppFlow

C. AWS Glue Data Catalog

D. Amazon Kinesis

Explanation:

The correct answer is B: Amazon AppFlow.

Here's why Amazon AppFlow is the best choice for this scenario:

1. Purpose and Functionality: Amazon AppFlow is a fully managed integration service that enables you to securely transfer data between Software as a Service (SaaS) applications and AWS services like Amazon S3, without needing to write custom code. It is specifically designed to facilitate the movement of data from external sources to AWS, which aligns perfectly with the company's need to gather data from third-party SaaS applications.

2. Ease of Use: Amazon AppFlow is designed to reduce operational overhead. It provides a user-friendly interface where you can set up data flows without managing any infrastructure. This means you can quickly configure data transfers with minimal setup, which is ideal for companies looking to save on operational tasks.

3. Security and Compliance: AppFlow automatically encrypts data in transit and at rest, and it allows you to configure data flows that meet compliance requirements. This ensures that the data transferred from the SaaS applications to Amazon S3 remains secure.

4. Data Transformation and Enrichment: AppFlow supports data transformation and filtering during the transfer process, which can help in preparing data for analytics in Amazon Redshift. This reduces the need for additional processing steps after the data is stored in S3.

5. Integration with AWS Services: AppFlow integrates seamlessly with Amazon S3, which is the target storage service in this scenario. It allows you to directly specify S3 buckets as destinations for the data flows, making it straightforward to set up and manage these transfers.

In contrast, the other options involve more operational complexity or are not directly suited for this task: - Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a managed service for Apache Kafka, which is more suitable for streaming data applications rather than batch transfers from SaaS applications. - AWS Glue Data Catalog is a metadata repository that helps organize data in S3 but is not responsible for moving data from SaaS applications. - Amazon Kinesis is a platform for streaming data, which again focuses on real-time data processing rather than the batch data ingestion from SaaS applications that AppFlow is tailored for.

Therefore, Amazon AppFlow provides the least operational overhead for integrating data from SaaS applications into Amazon S3 and subsequently using it for analytics in Amazon Redshift.

Question # 48

Date: February 02, 2024

A data engineer is using Amazon Athena to analyze sales data that is in Amazon S3. The data engineer writes a query to retrieve sales amounts for 2023 for several products from a table named sales_data. However, the query does not return results for all of the products that are in the sales_data table. The data engineer needs to troubleshoot the query to resolve the issue.

The data engineer's original query is as follows:

SELECT product_name, sum(sales_amount)

FROM sales_data -

WHERE year = 2023 -

GROUP BY product_name -

How should the data engineer modify the Athena query to meet these requirements?

A. Replace sum(sales_amount) with count(*) for the aggregation.
B. Change WHERE year = 2023 to WHERE extract(year FROM sales_data) = 2023.
C. Add HAVING sum(sales_amount) > 0 after the GROUP BY clause.
D. Remove the GROUP BY clause.

Correct Answer:

A. Replace sum(sales_amount) with count(*) for the aggregation.

B. Change WHERE year = 2023 to WHERE extract(year FROM sales_data) = 2023.

C. Add HAVING sum(sales_amount) > 0 after the GROUP BY clause.

D. Remove the GROUP BY clause.

Explanation:

The issue the data engineer is facing likely stems from the condition used in the WHERE clause to filter the sales data by year. The original query uses `WHERE year = 2023`, which assumes that there is a column named `year` in the `sales_data` table. If the `year` information is not stored in a dedicated `year` column but instead is part of a datetime or timestamp field within the sales data, this would result in the query not returning the expected results.

The correct answer, option B, suggests modifying the WHERE clause to `WHERE extract(year FROM sales_data) = 2023`. This change implies the use of the `EXTRACT` function, which is commonly used to retrieve a specific part (such as the year) from a date or timestamp field. By doing so, the query dynamically extracts the year from a datetime column in the `sales_data` table, ensuring that the condition accurately filters the data for the year 2023.

This modification solves the problem by correctly identifying sales records from the year 2023, assuming the date information is embedded within a datetime field rather than a standalone year column. Consequently, the query will now consider the correct subset of data, thus returning the expected results for the specified products. This approach is essential when the year is stored as part of a larger date field, which is a common scenario in sales data stored in databases or data lakes like Amazon S3.

Question # 49

Date: February 02, 2024

A security company stores IoT data that is in JSON format in an Amazon S3 bucket. The data structure can change when the company upgrades the IoT devices. The company wants to create a data catalog that includes the IoT data. The company's analytics department will use the data catalog to index the data.

Which solution will meet these requirements MOST cost-effectively?

A. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.
B. Create an Amazon Redshift provisioned cluster. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.
C. Create an Amazon Athena workgroup. Explore the data that is in Amazon S3 by using Apache Spark through Athena. Provide the Athena workgroup schema and tables to the analytics department.
D. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Correct Answer:

A. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

B. Create an Amazon Redshift provisioned cluster. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.

C. Create an Amazon Athena workgroup. Explore the data that is in Amazon S3 by using Apache Spark through Athena. Provide the Athena workgroup schema and tables to the analytics department.

D. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Explanation:

The correct answer is A. Let's break down why this solution is the most cost-effective and appropriate for the scenario described:

1. AWS Glue Data Catalog: This is a fully managed service that provides a central metadata repository for all your data assets, making it easier to manage and query data. It's particularly useful for data stored in Amazon S3 as it automatically discovers and catalogs data with minimal setup. This fits well with the requirement of creating a data catalog for IoT data stored in JSON format on S3.

2. AWS Glue Schema Registry: The schema registry is part of AWS Glue and allows you to define and enforce schemas for your streaming data. It supports schema evolution, which is crucial for IoT data that may change structure when devices are upgraded. This ensures that any changes in data structure are handled smoothly without breaking downstream processes.

3. AWS Glue Workload for Ingestion into Amazon Redshift Serverless: AWS Glue can orchestrate ETL (Extract, Transform, Load) jobs to move data into Amazon Redshift Serverless. Redshift Serverless is a cost-effective way to run analytics without managing clusters, as it automatically provisions and scales resources based on workload demands. This is ideal for the company's analytics department to use the data without the overhead of managing infrastructure.

4. Cost-Effectiveness: Option A leverages serverless services like AWS Glue and Redshift Serverless, which are more cost-effective than maintaining a provisioned Amazon Redshift cluster as in option B. Serverless services automatically scale and you pay only for what you use, reducing costs in scenarios where data volume or processing needs fluctuate.

In comparison to other options:

- Option B involves a provisioned Redshift cluster, which could be more expensive due to constant provisioning of resources regardless of usage. - Option C uses Amazon Athena and Apache Spark, which could be a viable solution, but it does not include a clear path for schema evolution or a robust data cataloging system like AWS Glue. - Option D is more complex and involves multiple AWS services, which could increase operational overhead and costs.

Therefore, option A offers a solution that effectively meets the company's requirements for flexibility, cost efficiency, and ease of managing evolving data structures.

Question # 50

Date: February 02, 2024

A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views.

Which solution will meet this requirement with the LEAST effort?

A. Use Apache Airflow to refresh the materialized views.
B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.
C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.
D. Use an AWS Glue workflow to refresh the materialized views.

Correct Answer:

A. Use Apache Airflow to refresh the materialized views.

B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.

C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.

D. Use an AWS Glue workflow to refresh the materialized views.

Explanation:

The question is about automating the refresh schedules for Amazon Redshift materialized views. Let's break down why option C is the correct answer with the least effort involved:

Option C: Use the query editor v2 in Amazon Redshift to refresh the materialized views. - Amazon Redshift's query editor v2 is a web-based tool that provides an easy way to work with your data in Redshift. It allows users to run SQL queries directly in their browser without needing to set up a separate client application. - By using query editor v2, you can create and schedule SQL queries, including those that refresh materialized views, directly within the Amazon Redshift environment. This tool provides a simple and integrated way to manage database operations, including the automation of tasks like refreshing materialized views. - The "least effort" aspect comes from the fact that using query editor v2 doesn't require additional setup or integration with other services. It leverages the existing capabilities of Amazon Redshift, making it a straightforward solution for this requirement.

Now, let's look at why the other options are less suitable:

Option A: Use Apache Airflow to refresh the materialized views. - Apache Airflow is a platform to programmatically author, schedule, and monitor workflows. While powerful, it requires setting up and maintaining an Airflow environment, which adds complexity and management overhead. This makes it less suitable for a solution requiring "least effort."

Option B: Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views. - AWS Lambda UDFs can be used to extend Redshift's capabilities by executing external code, but they are typically used for data processing tasks rather than straightforward SQL operations like refreshing materialized views. Additionally, setting up Lambda UDFs involves more configuration and is not the most direct approach for this task.

Option D: Use an AWS Glue workflow to refresh the materialized views. - AWS Glue is primarily a data integration service used for ETL (Extract, Transform, Load) processes. While you can use Glue workflows to orchestrate tasks, it involves creating and managing Glue jobs and workflows, which adds more complexity compared to using Redshift's built-in tools.

In summary, option C is the simplest and most direct solution, as it allows you to use Amazon Redshift's integrated query editor to automate the refresh of materialized views without the need for additional services or complex configurations.

Question # 51

Date: February 02, 2024

A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3.

The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically.
B. Use the AWS Glue Data Catalog as the central metadata repository. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata changes. Schedule the crawlers to run periodically to update the metadata catalog.
C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically.
D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

Correct Answer:

A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically.

B. Use the AWS Glue Data Catalog as the central metadata repository. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata changes. Schedule the crawlers to run periodically to update the metadata catalog.

C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically.

D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

Explanation:

The correct answer is B, which involves using the AWS Glue Data Catalog and AWS Glue crawlers. Let's break down why this is the most suitable solution with the least operational overhead for the given requirements:

1. AWS Glue Data Catalog: It is a fully managed service that acts as a central repository to store and manage metadata for data sources. It's designed specifically for use cases like the one described, where you need to maintain and manage metadata across various data stores such as Amazon RDS, Amazon Redshift, and files in Amazon S3.

2. AWS Glue Crawlers: Crawlers are an integral part of AWS Glue that can automatically connect to the data stores, infer the schema, and populate the Data Catalog with metadata. They are capable of detecting changes in the source metadata, which means they can automatically update the Data Catalog when there are changes in the data structure or new data is added.

3. Operational Overhead: Using AWS Glue and its crawlers involves minimal operational overhead. Once set up, crawlers can automatically run on a schedule to update the Data Catalog, ensuring that the metadata is always current without requiring manual intervention. This automation is a significant advantage over other solutions that might require more manual setup or maintenance.

4. Support for Multiple Data Types: AWS Glue can handle various data types, including structured data from databases like Amazon RDS and Amazon Redshift, as well as semi-structured data such as JSON and XML files stored in Amazon S3. This versatility makes it ideal for environments with diverse data sources.

5. Cost and Complexity: Compared to setting up custom solutions using Amazon Aurora or Amazon DynamoDB along with Lambda functions (as suggested in options A and C), AWS Glue provides a more straightforward, cost-effective, and scalable solution. It avoids the need to build and maintain custom code for metadata extraction and catalog updates.

Option D, while also utilizing the AWS Glue Data Catalog, suggests a more manual setup for structured data sources, which could lead to increased operational complexity compared to using fully automated crawlers for all sources as in Option B.

Overall, using AWS Glue Data Catalog and Glue crawlers (Option B) provides a seamless, automated, and low-maintenance solution that meets the company's requirements efficiently.

Question # 52

Date: February 02, 2024

A company is planning to migrate on-premises Apache Hadoop clusters to Amazon EMR. The company also needs to migrate a data catalog into a persistent storage solution.

The company currently stores the data catalog in an on-premises Apache Hive metastore on the Hadoop clusters. The company requires a serverless solution to migrate the data catalog.

Which solution will meet these requirements MOST cost-effectively?

A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.
B. Configure a Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.
C. Configure an external Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use Amazon Aurora MySQL to store the company's data catalog.
D. Configure a new Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use the new metastore as the company's data catalog.

Correct Answer:

A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.

B. Configure a Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.

C. Configure an external Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use Amazon Aurora MySQL to store the company's data catalog.

D. Configure a new Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use the new metastore as the company's data catalog.

Explanation:

The correct answer is B: Configure a Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.

Here’s why this solution is the most cost-effective and meets the company's requirements:

1. Serverless Solution: The company requires a serverless solution for the data catalog. AWS Glue Data Catalog is a fully managed, serverless service that provides a persistent data catalog, which perfectly matches this requirement. By using AWS Glue, the company avoids the need to manage any underlying infrastructure for the data catalog, which aligns with their need for a serverless solution.

2. Cost-Effectiveness: AWS Glue Data Catalog is cost-effective because it charges based on the number of objects stored and the number of requests made, rather than requiring dedicated infrastructure or instance management. This pay-as-you-go model is typically more cost-efficient for data cataloging compared to maintaining a self-managed service.

3. Integration with EMR: By configuring a Hive metastore in Amazon EMR and migrating the existing on-premises Hive metastore to it, the company can seamlessly integrate their existing setup into AWS without significant reconfiguration. Amazon EMR can work with the AWS Glue Data Catalog as an external metastore, allowing the company to utilize their existing Hive data catalog seamlessly in the cloud environment.

4. Preservation of Existing Metadata and Schema: Migrating the existing Hive metastore to Amazon EMR ensures that the company's current metadata and schema definitions are preserved. This minimizes the risk of data loss or the need for extensive reconfiguration, which can be both costly and time-consuming.

5. Avoiding Additional Database Management: Option B avoids the need for additional database management that would be required in options A and C, which mention using Amazon S3 and Amazon Aurora MySQL, respectively. These options introduce more complexity and management overhead, which contradicts the company's need for a simple, serverless, and cost-effective solution.

In summary, option B provides a direct path to leverage AWS services (Amazon EMR and AWS Glue) to meet the company's requirements for migrating their on-premises Hive metastore to a serverless and cost-effective solution, while maintaining compatibility and ease of management.

Question # 53

Date: February 02, 2024

A company stores data from an application in an Amazon DynamoDB table that operates in provisioned capacity mode. The workloads of the application have predictable throughput load on a regular schedule. Every Monday, there is an immediate increase in activity early in the morning. The application has very low usage during weekends.

The company must ensure that the application performs consistently during peak usage times.

Which solution will meet these requirements in the MOST cost-effective way?

A. Increase the provisioned capacity to the maximum capacity that is currently present during peak load times.
B. Divide the table into two tables. Provision each table with half of the provisioned capacity of the original table. Spread queries evenly across both tables.
C. Use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times. Schedule lower capacity during off-peak times.
D. Change the capacity mode from provisioned to on-demand. Configure the table to scale up and scale down based on the load on the table.

Correct Answer:

A. Increase the provisioned capacity to the maximum capacity that is currently present during peak load times.

B. Divide the table into two tables. Provision each table with half of the provisioned capacity of the original table. Spread queries evenly across both tables.

C. Use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times. Schedule lower capacity during off-peak times.

D. Change the capacity mode from provisioned to on-demand. Configure the table to scale up and scale down based on the load on the table.

Explanation:

The correct answer is C: Use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times. Schedule lower capacity during off-peak times.

Here's why this is the most cost-effective solution for the scenario:

1. Understanding Provisioned Capacity Mode: In Amazon DynamoDB, provisioned capacity mode allows you to specify the number of read and write capacity units required for your application. This mode is suitable when you can accurately predict your application's workload and want to optimize costs by not over-provisioning resources.

2. Predictable Workload: The scenario describes a predictable pattern of usage where there is a spike in activity every Monday morning and very low usage during weekends. This predictable pattern is ideal for scheduled scaling.

3. AWS Application Auto Scaling: This service allows you to automatically adjust the provisioned capacity of your DynamoDB tables according to a schedule. By using scheduled scaling, you can increase the provisioned capacity just before the expected spike on Monday mornings and decrease it during the weekends when usage is low. This ensures that your application has the necessary resources during peak times without incurring unnecessary costs during off-peak times.

4. Cost-Effectiveness: Option C is cost-effective because it avoids the need to maintain maximum capacity at all times (as suggested in option A). Instead, it allows you to dynamically scale capacity based on actual usage patterns, thereby saving money during periods of low demand.

5. Drawbacks of Other Options: - Option A: Increasing the provisioned capacity to the maximum required at peak times means you will be paying for this maximum capacity even when it's not needed (e.g., during weekends), which is not cost-effective. - Option B: Splitting the table into two would add complexity without necessarily addressing the cost issue. It doesn't leverage the predictable workload pattern effectively. - Option D: Switching to on-demand capacity mode could be more expensive than provisioned capacity if your application has high and predictable usage patterns. On-demand capacity is better suited for unpredictable workloads.

By using scheduled scaling with AWS Application Auto Scaling, you can ensure consistent performance during peak times while optimizing costs based on your application's predictable usage patterns.

Question # 54

Date: February 02, 2024

A company uses an Amazon Redshift provisioned cluster as its database. The Redshift cluster has five reserved ra3.4xlarge nodes and uses key distribution.

A data engineer notices that one of the nodes frequently has a CPU load over 90%. SQL Queries that run on the node are queued. The other four nodes usually have a CPU load under 15% during daily operations.

The data engineer wants to maintain the current number of compute nodes. The data engineer also wants to balance the load more evenly across all five compute nodes.

Which solution will meet these requirements?

A. Change the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.
B. Change the distribution key to the table column that has the largest dimension.
C. Upgrade the reserved node from ra3.4xlarge to ra3.16xlarge.
D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Correct Answer:

A. Change the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

B. Change the distribution key to the table column that has the largest dimension.

C. Upgrade the reserved node from ra3.4xlarge to ra3.16xlarge.

D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Explanation:

The correct answer is B: Change the distribution key to the table column that has the largest dimension.

In Amazon Redshift, the distribution key determines how data is distributed across the nodes in the cluster. Properly distributing data can significantly impact query performance and resource utilization. Here's why changing the distribution key to the table column with the largest dimension is the correct choice:

1. Understanding Distribution Keys: In Redshift, the distribution key is used to distribute data across different nodes evenly. When a column with high cardinality (many unique values) is used as a distribution key, it helps in spreading the data more evenly across all nodes. This ensures that no single node is overloaded while others remain underutilized.

2. Current Issue: The problem described is that one node is experiencing a much higher CPU load compared to others. This suggests an imbalance in how data is distributed. If a particular node is handling a significant portion of the data, it will be overworked, leading to high CPU usage and queued queries.

3. Balancing Load: By changing the distribution key to a column with the largest dimension, which typically means it has high cardinality, you are likely to achieve a more balanced distribution of data. This means that the queries will also be distributed more evenly across nodes, reducing the load imbalance and improving overall performance.

4. Why Other Options Are Less Suitable: - Option A (Changing the sort key): Sort keys optimize query performance by allowing Redshift to quickly locate the data it needs, but they do not affect how data is distributed across nodes. They won't solve the problem of one node being overutilized. - Option C (Upgrading the node type): While upgrading would increase the capacity of the overloaded node, it doesn't address the root cause of data imbalance. It also contradicts the requirement to maintain the current number of compute nodes. - Option D (Changing the primary key): Primary keys in Redshift are informational only and do not enforce uniqueness or impact data distribution. Changing the primary key will not solve the distribution issue.

In summary, the primary goal is to balance the load by distributing data more evenly across nodes, and changing the distribution key to a column with high cardinality is the best way to achieve this in Amazon Redshift.

Question # 55

Date: February 02, 2024

A company needs to build a data lake in AWS. The company must provide row-level data access and column-level data access to specific teams. The teams will access the data by using Amazon Athena, Amazon Redshift Spectrum, and Apache Hive from Amazon EMR.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon S3 for data lake storage. Use S3 access policies to restrict data access by rows and columns. Provide data access through Amazon S3.
B. Use Amazon S3 for data lake storage. Use Apache Ranger through Amazon EMR to restrict data access by rows and columns. Provide data access by using Apache Pig.
C. Use Amazon Redshift for data lake storage. Use Redshift security policies to restrict data access by rows and columns. Provide data access by using Apache Spark and Amazon Athena federated queries.
D. Use Amazon S3 for data lake storage. Use AWS Lake Formation to restrict data access by rows and columns. Provide data access through AWS Lake Formation.

Correct Answer:

A. Use Amazon S3 for data lake storage. Use S3 access policies to restrict data access by rows and columns. Provide data access through Amazon S3.

B. Use Amazon S3 for data lake storage. Use Apache Ranger through Amazon EMR to restrict data access by rows and columns. Provide data access by using Apache Pig.

C. Use Amazon Redshift for data lake storage. Use Redshift security policies to restrict data access by rows and columns. Provide data access by using Apache Spark and Amazon Athena federated queries.

D. Use Amazon S3 for data lake storage. Use AWS Lake Formation to restrict data access by rows and columns. Provide data access through AWS Lake Formation.

Explanation:

The correct answer is D: Use Amazon S3 for data lake storage. Use AWS Lake Formation to restrict data access by rows and columns. Provide data access through AWS Lake Formation.

Here's why this solution is the best choice with the least operational overhead for the company's requirements:

1. Data Storage with Amazon S3: Amazon S3 is a highly scalable and durable storage solution, ideal for storing vast amounts of data, which is typical for data lakes. It provides cost-effective storage and integrates seamlessly with various AWS services.

2. AWS Lake Formation: This service is specifically designed to simplify the process of setting up a secure data lake. It enables fine-grained access control, allowing you to manage both row-level and column-level permissions efficiently. Lake Formation abstracts and automates much of the complexity involved in setting up a data lake, significantly reducing operational overhead compared to more manual approaches.

3. Access Management: AWS Lake Formation provides a centralized and simplified way to manage data access permissions. It integrates with AWS Identity and Access Management (IAM) and allows you to define and enforce policies that grant users specific access to data, ensuring that only authorized users can view or query certain data rows and columns.

4. Integration with Data Analytics Services: Lake Formation seamlessly integrates with Amazon Athena, Amazon Redshift Spectrum, and Apache Hive on Amazon EMR. This integration allows users to query and analyze data stored in S3 using these services without additional complex setup, maintaining data security and access controls defined in Lake Formation.

5. Least Operational Overhead: Compared to other options, using Lake Formation requires less manual configuration and ongoing maintenance. For example, manually setting S3 access policies (Option A) or using Apache Ranger with Amazon EMR (Option B) involves more complexity and potential for errors. These approaches typically require more in-depth knowledge and management effort to ensure security policies are correctly set and maintained.

In summary, Option D provides a streamlined, efficient, and secure method for managing access to a data lake in AWS. It leverages AWS Lake Formation's capabilities to minimize the complexity and operational overhead typically associated with setting up and managing access controls in a data lake environment.

Question # 56

Date: February 03, 2024

A data engineer must build an extract, transform, and load (ETL) pipeline to process and load data from 10 source systems into 10 tables that are in an Amazon Redshift database. All the source systems generate .csv, JSON, or Apache Parquet files every 15 minutes. The source systems all deliver files into one Amazon S3 bucket. The file sizes range from 10 MB to 20 GB. The ETL pipeline must function correctly despite changes to the data schema.

Which data pipeline solutions will meet these requirements? (Choose two.)

A. Use an Amazon EventBridge rule to run an AWS Glue job every 15 minutes. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
B. Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minutes. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
C. Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket. Configure an AWS Glue job to process and load the data into the Amazon Redshift tables. Create a second Lambda function to run the AWS Glue job. Create an Amazon EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.
D. Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
E. Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFrame. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream. Configure the delivery stream to load data into the Amazon Redshift tables.

Correct Answer:

A. Use an Amazon EventBridge rule to run an AWS Glue job every 15 minutes. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

B. Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minutes. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

C. Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket. Configure an AWS Glue job to process and load the data into the Amazon Redshift tables. Create a second Lambda function to run the AWS Glue job. Create an Amazon EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.

D. Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

E. Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFrame. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream. Configure the delivery stream to load data into the Amazon Redshift tables.

Explanation:

The correct answers to the question are B and D. Let me explain why these solutions meet the requirements for building an ETL pipeline to process and load data into Amazon Redshift from multiple source systems.

1. Option B Explanation: - Amazon EventBridge Rule and AWS Glue Workflow: This solution uses an Amazon EventBridge rule to invoke an AWS Glue workflow every 15 minutes. The workflow consists of a sequence of tasks, including running an AWS Glue crawler followed by an AWS Glue job. - AWS Glue Crawler: The crawler is crucial because it automatically infers the schema of the data in the S3 bucket. This feature is particularly important given the requirement that the ETL pipeline must adapt to schema changes in the source data. The crawler's ability to detect new or altered schemas ensures that the subsequent ETL process can handle these changes without manual intervention. - AWS Glue Job: After the crawler updates the schema, the AWS Glue job processes the data and loads it into the Amazon Redshift tables. This job uses the metadata created by the crawler to correctly interpret the data structure. - Automatic Triggering: The use of EventBridge to schedule the workflow ensures that the process starts reliably every 15 minutes, aligning with the frequency of data generation by the source systems.

2. Option D Explanation: - AWS Lambda Function and AWS Glue Workflow: This solution sets up an AWS Lambda function to trigger an AWS Glue workflow as soon as a new file is loaded into the S3 bucket. This event-driven approach ensures immediate processing of new data. - Schema Adaptability: Similar to Option B, this workflow includes an AWS Glue crawler followed by a Glue job. This combination allows the system to handle schema changes dynamically. The crawler updates the data catalog with any schema adjustments, which the Glue job then uses to correctly process and load data. - Real-Time Triggering: By using a Lambda function to kick off the workflow on file arrival, this solution offers real-time responsiveness to data ingestion events, providing a continuous and efficient data processing pipeline.

Both solutions leverage AWS Glue's capabilities to adapt to schema changes, which is a critical requirement. They ensure that the pipeline remains robust and flexible, automatically adjusting to changes in the input data structure without manual intervention. Additionally, these solutions provide timely data processing by using EventBridge for scheduled execution or Lambda for event-driven execution, aligning with the data generation and delivery schedule of the source systems.

Question # 57

Date: February 02, 2024

A company stores details about transactions in an Amazon S3 bucket. The company wants to log all writes to the S3 bucket into another S3 bucket that is in the same AWS Region.

Which solution will meet this requirement with the LEAST operational effort?

A. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function. Program the Lambda function to write the event to Amazon Kinesis Data Firehose. Configure Kinesis Data Firehose to write the event to the logs S3 bucket.
B. Create a trail of management events in AWS CloudTraiL. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.
C. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function. Program the Lambda function to write the events to the logs S3 bucket.
D. Create a trail of data events in AWS CloudTraiL. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.

Correct Answer:

A. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function. Program the Lambda function to write the event to Amazon Kinesis Data Firehose. Configure Kinesis Data Firehose to write the event to the logs S3 bucket.

B. Create a trail of management events in AWS CloudTraiL. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.

C. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function. Program the Lambda function to write the events to the logs S3 bucket.

D. Create a trail of data events in AWS CloudTraiL. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.

Explanation:

The correct answer is D: Create a trail of data events in AWS CloudTrail. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.

Here's the detailed explanation:

1. CloudTrail Data Events: AWS CloudTrail allows you to set up trails to log data events for S3 buckets. Data events provide detailed information about the resource operations performed on or within a resource, such as an Amazon S3 bucket. These events are different from management events, which log management operations and activities related to AWS resources.

2. Focus on Data Events: In this scenario, the company is interested in logging specific actions—specifically, write operations—on an S3 bucket. Data events are tailored for this purpose, as they can track object-level API activity, such as PutObject, which involves writing data to a bucket.

3. Minimal Operational Effort: Using CloudTrail to log data events requires minimal operational effort compared to manually setting up and maintaining other services like AWS Lambda or Amazon Kinesis Data Firehose. Once set up, CloudTrail automatically logs the specified events and delivers them to the specified S3 bucket.

4. Configuration Specifics: By configuring the trail to specify an empty prefix and selecting write-only events, you ensure that the trail captures all write operations to the S3 bucket. The empty prefix means every object write operation, regardless of its key, is logged. The write-only events filter ensures you're only capturing the relevant operations, thus saving on storage and processing costs.

5. Direct Logging to S3: By specifying the logs S3 bucket as the destination, CloudTrail will automatically deliver the log files to this bucket without additional processing or intervention. This setup simplifies the architecture and reduces the need for ongoing management.

6. Alternative Options: The other options involve more components, such as AWS Lambda and Kinesis Data Firehose, which add complexity and operational overhead. These solutions would require custom code and potentially more resources to manage and maintain, which is not as efficient for this use case as the straightforward CloudTrail configuration.

In summary, using AWS CloudTrail to log data events for write operations on an S3 bucket is an efficient, low-maintenance solution to meet the requirement of logging all writes to the S3 bucket. This method leverages AWS's built-in capabilities to automatically and securely handle the logging process with minimal configuration and operational burden.

Question # 58

Date: February 02, 2024

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

Which solution will meet these requirements with the LEAST development effort?

A. Use Amazon EMR and Apache Ranger.
B. Use a Hive metastore on an EMR cluster.
C. Use the AWS Glue Data Catalog.
D. Use a metastore on an Amazon RDS for MySQL DB instance.

Correct Answer:

A. Use Amazon EMR and Apache Ranger.

B. Use a Hive metastore on an EMR cluster.

C. Use the AWS Glue Data Catalog.

D. Use a metastore on an Amazon RDS for MySQL DB instance.

Explanation:

The correct answer is C: Use the AWS Glue Data Catalog.

Here's why this is the best solution with the least development effort:

1. Centralized Metadata Management: AWS Glue Data Catalog is a fully managed service that provides a centralized metadata repository. It is designed specifically for storing, annotating, and sharing metadata across various AWS services, including Amazon EMR and Amazon Athena. This makes it a natural fit for the requirement of maintaining a central metadata repository.

2. Integration with AWS Services: AWS Glue Data Catalog integrates seamlessly with Amazon EMR and Amazon Athena. This allows you to easily query and process data using these services without additional integration work. The Data Catalog acts as a common metadata repository that both services can access directly.

3. Ease of Migration: AWS Glue provides built-in tools to import metadata from Apache Hive. This feature simplifies the migration process from a Hive metastore to the AWS Glue Data Catalog. The data engineer can use AWS Glue's "Crawler" functionality to automatically discover and catalog metadata, minimizing manual effort.

4. Minimal Development Effort: Since AWS Glue Data Catalog is a managed service, it requires less setup and maintenance compared to other options that might involve more manual configuration or infrastructure management. This reduces the development effort significantly, as there's no need to manage servers or databases manually.

5. Scalability and Reliability: Being a managed service, AWS Glue Data Catalog is designed to scale automatically and provide high availability. This means that as the data and metadata grow, the service can handle the increased load without requiring manual intervention.

In contrast, the other options would require more setup and maintenance:

- Option A (Amazon EMR and Apache Ranger): While Apache Ranger can manage metadata, it's more focused on security and access control rather than being a central metadata repository. It also requires more complex setup and integration work.

- Option B (Hive metastore on an EMR cluster): This could work, but it would require maintaining a separate Hive metastore within the EMR environment, which involves more management overhead compared to using a managed service like AWS Glue.

- Option D (Metastore on an Amazon RDS for MySQL DB instance): This would require setting up and maintaining a MySQL database, which involves additional administration tasks, such as scaling, backup, and recovery. It also doesn't provide the same level of built-in integration with AWS services as the Glue Data Catalog.

Overall, AWS Glue Data Catalog offers the simplest, most seamless solution with the least development effort for integrating metadata from Hive and supporting queries in both Amazon EMR and Athena.

Question # 59

Date: February 02, 2024

A company's data engineer needs to optimize the performance of table SQL queries. The company stores data in an Amazon Redshift cluster. The data engineer cannot increase the size of the cluster because of budget constraints.

The company stores the data in multiple tables and loads the data by using the EVEN distribution style. Some tables are hundreds of gigabytes in size. Other tables are less than 10 MB in size.

Which solution will meet these requirements?

A. Keep using the EVEN distribution style for all tables. Specify primary and foreign keys for all tables.
B. Use the ALL distribution style for large tables. Specify primary and foreign keys for all tables.
C. Use the ALL distribution style for rarely updated small tables. Specify primary and foreign keys for all tables.
D. Specify a combination of distribution, sort, and partition keys for all tables.

Correct Answer:

A. Keep using the EVEN distribution style for all tables. Specify primary and foreign keys for all tables.

B. Use the ALL distribution style for large tables. Specify primary and foreign keys for all tables.

C. Use the ALL distribution style for rarely updated small tables. Specify primary and foreign keys for all tables.

D. Specify a combination of distribution, sort, and partition keys for all tables.

Explanation:

When working with Amazon Redshift, optimizing the performance of SQL queries without increasing the cluster size involves making strategic choices about how data is distributed across nodes. The distribution style determines how data is spread across the nodes in a Redshift cluster and can significantly impact query performance, particularly for join operations.

Let's break down the provided options and see why option C is the most suitable choice given the constraints:

1. EVEN Distribution Style: This is the default distribution style in Redshift, where data is spread evenly across all nodes. While this can ensure balanced storage and compute usage, it might not be optimal for query performance, especially if tables are joined frequently. The EVEN distribution does not necessarily minimize data movement between nodes, which is a key factor for query performance.

2. ALL Distribution Style for Small Tables: The ALL distribution style replicates the entire table on each node. This can be a great choice for small tables that are frequently joined with larger tables, as it eliminates the need for data movement across nodes during joins. Since the small tables in the scenario are less than 10 MB, using the ALL distribution style is feasible without consuming excessive storage space. This approach can significantly enhance query performance for joins involving these small tables.

3. Primary and Foreign Keys: Specifying primary and foreign keys is important for query optimization, even though Redshift does not enforce these constraints. They provide valuable metadata that the query planner can use to optimize execution plans. While they alone don't affect data distribution, they contribute to better query execution when combined with optimal distribution styles.

4. Large Tables and Rarely Updated Small Tables: In the scenario, applying the ALL distribution style is specifically mentioned for "rarely updated small tables." This is because ALL distribution could increase the overhead of updates and inserts due to the need to replicate changes across all nodes. However, for small, infrequently updated tables, this overhead is minimal, making this strategy effective.

5. Option D: While specifying a combination of distribution, sort, and partition keys could optimize queries, the question specifically restricts increasing the cluster size, and such an approach might not address the core issue of minimizing cross-node traffic as effectively as using the ALL distribution style for small tables.

Therefore, option C is correct because it suggests using the ALL distribution style for small tables that are rarely updated, which optimizes join performance by minimizing data movement. Additionally, specifying primary and foreign keys helps the query planner optimize execution, aligning well with the constraints and requirements of the scenario.

Question # 60

Date: February 02, 2024

A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during non-business hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to comply with company policies.

A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.

Which solution will meet these requirements with the LEAST operational overhead?

A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day.
B. Use the query result reuse feature of Amazon Athena for the SQL queries.
C. Add an Amazon ElastiCache cluster between the BI application and Athena.
D. Change the format of the files that are in the dataset to Apache Parquet.

Correct Answer:

A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day.

B. Use the query result reuse feature of Amazon Athena for the SQL queries.

C. Add an Amazon ElastiCache cluster between the BI application and Athena.

D. Change the format of the files that are in the dataset to Apache Parquet.

Explanation:

Certainly! Let's delve into the details of why option B, "Use the query result reuse feature of Amazon Athena for the SQL queries," is the most suitable choice for cost optimization with the least operational overhead.

Amazon Athena is a serverless query service that allows you to run SQL queries on data stored in Amazon S3. It charges based on the amount of data scanned by your queries. Therefore, minimizing the amount of data scanned can significantly reduce costs.

Here's why option B is the best fit:

1. Query Result Reuse: Athena's query result reuse feature allows you to save and reuse the results of previous queries. If the same query is run multiple times and the data hasn't changed, Athena can return the cached results instead of reprocessing the query from scratch. This reduces the amount of data scanned and thus lowers the cost. Since the dataset is updated only once a day by AWS Glue, using cached results for queries run within the same day ensures that you don't incur additional costs for re-scanning the same data.

2. Operational Overhead: Implementing query result reuse has minimal operational overhead. You don't need to manually manage cache servers or additional infrastructure, such as in-memory databases, making it a straightforward solution.

3. Cost Efficiency: Since the Athena query charges are based on the data scanned, reusing results drastically cuts down on costs, especially if the same or similar queries are frequently run within the day.

4. Compliance with Data Refresh Requirements: The BI application requires data refresh every hour, but the underlying data changes only once a day. By reusing query results, you can meet the hourly refresh needs without re-scanning the data unless it has been updated, aligning with company policies while optimizing costs.

Now, let's briefly examine why the other options are less ideal:

- Option A: Moving data to S3 Glacier Deep Archive would not be suitable because it is a cold storage class designed for long-term archival, not for active querying. It would result in high latency and retrieval costs, which are not aligned with the requirements of frequent data access for BI applications.

- Option C: Adding an Amazon ElastiCache cluster would introduce additional infrastructure and operational costs. It would also require managing the cache lifecycle and could increase complexity without necessarily addressing the fundamental cost concern of data scanning in Athena.

- Option D: Changing the file format to Apache Parquet could indeed reduce the amount of data scanned because Parquet is a columnar storage format that allows for more efficient queries. However, it involves a data transformation process, which could incur its own costs and operational overhead. While beneficial, it does not directly address the repeated query cost in the same way that result reuse does.

In summary, option B leverages Athena's built-in feature to optimize costs effectively with minimal operational intervention, making it the best choice given the scenario.

Question # 61

Date: February 02, 2024

A manufacturing company wants to collect data from sensors. A data engineer needs to implement a solution that ingests sensor data in near real time.

The solution must store the data to a persistent data store. The solution must store the data in nested JSON format. The company must have the ability to query from the data store with a latency of less than 10 milliseconds.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use a self-hosted Apache Kafka cluster to capture the sensor data. Store the data in Amazon S3 for querying.
B. Use AWS Lambda to process the sensor data. Store the data in Amazon S3 for querying.
C. Use Amazon Kinesis Data Streams to capture the sensor data. Store the data in Amazon DynamoDB for querying.
D. Use Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data. Use AWS Glue to store the data in Amazon RDS for querying.

Correct Answer:

A. Use a self-hosted Apache Kafka cluster to capture the sensor data. Store the data in Amazon S3 for querying.

B. Use AWS Lambda to process the sensor data. Store the data in Amazon S3 for querying.

C. Use Amazon Kinesis Data Streams to capture the sensor data. Store the data in Amazon DynamoDB for querying.

D. Use Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data. Use AWS Glue to store the data in Amazon RDS for querying.

Explanation:

The correct answer is C: Use Amazon Kinesis Data Streams to capture the sensor data. Store the data in Amazon DynamoDB for querying.

Here's why this solution is appropriate:

1. Near Real-Time Data Ingestion: - Amazon Kinesis Data Streams is designed for real-time data streaming. It can continuously capture and process large streams of data records in real-time. This makes it ideal for ingesting sensor data, which typically requires handling data that arrives continuously and needs to be processed with minimal latency.

2. Persistent Data Store: - Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. It can store the ingested data persistently and is highly durable, making it a suitable persistent data store.

3. Nested JSON Format: - DynamoDB supports storing data in JSON format, including nested JSON structures. This means it can naturally handle and store the JSON data format as required by the company.

4. Low Query Latency: - DynamoDB is optimized for high-speed read and write operations, and it can deliver single-digit millisecond performance. The requirement to query the data store with a latency of less than 10 milliseconds aligns well with DynamoDB's capabilities, ensuring that queries are performed quickly.

5. Least Operational Overhead: - Using AWS managed services like Kinesis and DynamoDB significantly reduces operational overhead. Both services are fully managed, meaning AWS takes care of the infrastructure management, scaling, and maintenance. This allows the data engineering team to focus on the data processing logic rather than on managing the infrastructure.

In contrast, other options involve higher operational complexity or do not meet the latency requirements:

- Option A suggests using a self-hosted Apache Kafka cluster, which requires significant setup and maintenance overhead, and storing data in S3, where querying would not meet the latency requirement. - Option B involves Lambda and S3, which, although serverless, would not provide the sub-10-millisecond query latency required. - Option D uses Amazon SQS and AWS Glue with Amazon RDS, which would introduce higher latency and complexity, particularly because RDS is more suitable for structured relational data and may not efficiently handle nested JSON with the required latency.

Thus, option C provides a comprehensive solution that efficiently meets all the specified requirements with minimal operational overhead.

Question # 62

Date: March 03, 2024

A company has used an Amazon Redshift table that is named Orders for 6 months. The company performs weekly updates and deletes on the table. The table has an interleaved sort key on a column that contains AWS Regions.

The company wants to reclaim disk space so that the company will not run out of storage space. The company also wants to analyze the sort key column.

Which Amazon Redshift command will meet these requirements?

A. VACUUM FULL Orders
B. VACUUM DELETE ONLY Orders
C. VACUUM REINDEX Orders
D. VACUUM SORT ONLY Orders

Correct Answer:

A. VACUUM FULL Orders

B. VACUUM DELETE ONLY Orders

C. VACUUM REINDEX Orders

D. VACUUM SORT ONLY Orders

Explanation:

The correct answer to this question is C: VACUUM REINDEX Orders.

To understand why this is the correct choice, let's break down the requirements and what each VACUUM operation does in Amazon Redshift:

1. Disk Space Reclamation: The company wants to reclaim disk space. Over time, as data is deleted or updated in a Redshift table, it can lead to fragmented and inefficient storage. This inefficiency is due to the nature of Redshift's columnar storage format, which doesn't automatically reclaim space from deleted or updated rows. Therefore, a VACUUM operation is typically required to clean up and compact the table.

2. Interleaved Sort Key: The table uses an interleaved sort key on a column containing AWS Regions. Interleaved sort keys are used in Redshift to optimize query performance on columns that are frequently filtered in queries. However, they can become suboptimal after substantial updates or deletes, as these operations can lead to a skewed sort order, especially if new data distributions don't match the original sorting.

3. Analyze the Sort Key Column: The company wants to analyze the sort key column. This suggests a need to ensure that the sort key is functioning effectively, which involves proper sorting and indexing.

Now, let's look at what each option does:

- A. VACUUM FULL Orders: This operation reclaims space and re-sorts the entire table. It's comprehensive but can be resource-intensive and time-consuming, especially for large tables. It doesn't specifically address interleaved sort keys' needs.

- B. VACUUM DELETE ONLY Orders: This operation focuses only on reclaiming space from deleted rows. While it helps with space, it doesn't re-sort the data, which is crucial for interleaved sort keys.

- C. VACUUM REINDEX Orders: This operation is specifically designed for tables with interleaved sort keys. It reindexes the interleaved sort key to improve query performance without the overhead of a full table re-sort. This makes it well-suited for situations where the interleaved sort key has become suboptimal due to updates and deletes.

- D. VACUUM SORT ONLY Orders: This operation re-sorts the table without reclaiming space from deleted rows. It doesn't specifically address interleaved sort key issues.

Given the requirements, VACUUM REINDEX (Option C) is the most appropriate command. It addresses both the need to optimize the interleaved sort key for better query performance and reclaims space that might be wasted due to outdated indexing, aligning perfectly with the company's objectives.

Question # 63

Date: February 02, 2024

During a security review, a company identified a vulnerability in an AWS Glue job. The company discovered that credentials to access an Amazon Redshift cluster were hard coded in the job script.

A data engineer must remediate the security vulnerability in the AWS Glue job. The solution must securely store the credentials.

Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

A. Store the credentials in the AWS Glue job parameters.
B. Store the credentials in a configuration file that is in an Amazon S3 bucket.
C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.
D. Store the credentials in AWS Secrets Manager.
E. Grant the AWS Glue job IAM role access to the stored credentials.

Correct Answer:

A. Store the credentials in the AWS Glue job parameters.

B. Store the credentials in a configuration file that is in an Amazon S3 bucket.

C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.

D. Store the credentials in AWS Secrets Manager.

E. Grant the AWS Glue job IAM role access to the stored credentials.

Explanation:

Certainly! Let's break down why options D and E are the correct steps to address the security vulnerability in the AWS Glue job:

Option D: Store the credentials in AWS Secrets Manager.

AWS Secrets Manager is a service designed specifically for managing sensitive information, such as database credentials, API keys, and other secrets needed by your applications. By storing your Amazon Redshift credentials in AWS Secrets Manager, you ensure that they are securely encrypted and managed according to best practices. AWS Secrets Manager helps in rotating, managing, and retrieving database credentials, API keys, and other secrets throughout their lifecycle. This practice not only enhances security by removing hard-coded credentials from your codebase but also makes it easier to update or rotate credentials without modifying the application code.

Option E: Grant the AWS Glue job IAM role access to the stored credentials.

After storing the credentials in AWS Secrets Manager, the AWS Glue job needs permission to access these credentials. This is where AWS Identity and Access Management (IAM) comes into play. By granting the AWS Glue job's IAM role permission to access the secrets stored in AWS Secrets Manager, you ensure that the job can securely retrieve the credentials at runtime without embedding them in the script. This is a critical step because, without the necessary permissions, the Glue job won't be able to access the credentials it needs to connect to the Amazon Redshift cluster.

Together, these two steps (D and E) provide a robust solution for managing sensitive credentials in AWS Glue jobs. They adhere to security best practices by avoiding hard-coded credentials and using AWS services designed for secure secret management and access control. This approach not only mitigates the identified security vulnerability but also fosters a more secure and maintainable infrastructure.

Question # 64

Date: February 02, 2024

A data engineer is configuring Amazon SageMaker Studio to use AWS Glue interactive sessions to prepare data for machine learning (ML) models.

The data engineer receives an access denied error when the data engineer tries to prepare the data by using SageMaker Studio.

Which change should the engineer make to gain access to SageMaker Studio?

A. Add the AWSGlueServiceRole managed policy to the data engineer's IAM user.
B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.
C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.
D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

Correct Answer:

A. Add the AWSGlueServiceRole managed policy to the data engineer's IAM user.

B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.

C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.

D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

Explanation:

To understand why option B is the correct answer, let's break down the situation and the role of each component involved:

1. Amazon SageMaker Studio: This is a web-based integrated development environment (IDE) for machine learning that allows data engineers and data scientists to build, train, and deploy ML models. It's a part of the SageMaker suite of services.

2. AWS Glue Interactive Sessions: These are used for data preparation tasks and allow users to interactively run data processing jobs on a serverless Spark cluster. AWS Glue is generally used for data transformation and preparation tasks.

3. IAM and Policies: AWS Identity and Access Management (IAM) is used to manage access to AWS services and resources securely. Access is controlled by defining policies that are attached to IAM identities (such as users, groups, or roles).

Now, let's focus on why the specific option B is the solution:

- sts:AssumeRole Action: This action allows an IAM user or service to assume a role defined in AWS. When you assume a role, you get temporary security credentials that take on the permissions of the assumed role. In this context, the data engineer needs to assume roles that allow interaction with both AWS Glue and SageMaker.

- Trust Policy: This is part of a role's settings that define which principals (users, groups, or services) can assume the role. The trust policy is essential for cross-service interactions where one service needs to interact with another using roles.

Explanation of Correct Choice (B):

- Role Assumption Requirement: To interact with AWS Glue and SageMaker, the data engineer needs to assume roles that have the necessary permissions for these services. This is because specific actions like starting Glue interactive sessions might require permissions that are different from just accessing SageMaker Studio.

- Action Required: By adding a policy that includes the `sts:AssumeRole` action, the data engineer is allowed to assume the roles necessary for interacting with these services. This permits the necessary cross-service access that SageMaker Studio needs to perform data preparation tasks using AWS Glue.

- Service Principals: The mention of service principals in the trust policy refers to allowing specific AWS services (like Glue and SageMaker) to be trusted entities that can assume the roles as needed.

The other options are not correct because:

- Option A: Adding `AWSGlueServiceRole` gives permissions directly related to Glue but doesn't address the cross-service role assumption needed for SageMaker Studio and Glue interaction.

- Option C: `AmazonSageMakerFullAccess` provides extensive permissions for SageMaker but does not cover the Glue-specific role assumption needed.

- Option D: `sts:AddAssociation` is not a valid AWS action for IAM policies and does not address the issue at hand.

Hence, the correct approach to resolve the access denied error is to configure the IAM user with the ability to assume the necessary roles for AWS Glue and SageMaker through the `sts:AssumeRole` action, as specified in option B.

Question # 65

Date: June 15, 2024

A company has a business intelligence platform on AWS. The company uses an AWS Storage Gateway Amazon S3 File Gateway to transfer files from the company's on-premises environment to an Amazon S3 bucket.

A data engineer needs to setup a process that will automatically launch an AWS Glue workflow to run a series of AWS Glue jobs when each file transfer finishes successfully.

Which solution will meet these requirements with the LEAST operational overhead?

A. Determine when the file transfers usually finish based on previous successful file transfers. Set up an Amazon EventBridge scheduled event to initiate the AWS Glue jobs at that time of day.
B. Set up an Amazon EventBridge event that initiates the AWS Glue workflow after every successful S3 File Gateway file transfer event.
C. Set up an on-demand AWS Glue workflow so that the data engineer can start the AWS Glue workflow when each file transfer is complete.
D. Set up an AWS Lambda function that will invoke the AWS Glue Workflow. Set up an event for the creation of an S3 object as a trigger for the Lambda function.

Correct Answer:

A. Determine when the file transfers usually finish based on previous successful file transfers. Set up an Amazon EventBridge scheduled event to initiate the AWS Glue jobs at that time of day.

B. Set up an Amazon EventBridge event that initiates the AWS Glue workflow after every successful S3 File Gateway file transfer event.

C. Set up an on-demand AWS Glue workflow so that the data engineer can start the AWS Glue workflow when each file transfer is complete.

D. Set up an AWS Lambda function that will invoke the AWS Glue Workflow. Set up an event for the creation of an S3 object as a trigger for the Lambda function.

Explanation:

The correct answer is B, which involves setting up an Amazon EventBridge event to initiate the AWS Glue workflow after every successful S3 File Gateway file transfer event. Let's break down why this is the best choice and has the least operational overhead:

1. Automatic Triggering: Option B leverages Amazon EventBridge to automatically detect events in your AWS environment. In this context, when a file is successfully transferred to the Amazon S3 bucket via the S3 File Gateway, EventBridge can capture this event and trigger the AWS Glue workflow. This automates the process, eliminating the need for manual intervention or complex scheduling logic.

2. Minimal Configuration: Setting up an EventBridge rule to trigger an AWS Glue workflow is straightforward. You define an event pattern that matches the specific S3 event (such as a file creation event in the bucket), and specify the target as the AWS Glue workflow. This setup is relatively simple and requires minimal ongoing management, which reduces operational overhead.

3. Real-time Processing: By using EventBridge, the workflow can be triggered in real-time as soon as the file transfer completes. This ensures that data processing can begin immediately, enhancing the efficiency and responsiveness of data pipelines.

4. Scalability and Reliability: EventBridge is a fully managed service that scales automatically and is highly reliable, ensuring that the detection of events and triggering of workflows happens seamlessly, even as the volume of file transfers increases.

5. No Need for Manual Scheduling or Monitoring: Unlike option A, which involves estimating transfer completion times and setting up a scheduled event, option B does not require predicting or monitoring when file transfers finish. EventBridge automatically handles this, reducing the potential for errors or delays due to incorrect timing assumptions.

6. Direct Integration: EventBridge has direct integration with AWS Glue, which simplifies the process of linking events to workflows. This direct integration ensures that the triggering of workflows is robust and less prone to misconfigurations.

In contrast, options A, C, and D either involve more manual intervention, require complex custom setups, or introduce additional components (like Lambda in option D) that increase the complexity and maintenance burden. Therefore, option B is the most efficient and streamlined solution for automatically launching AWS Glue workflows with the least operational overhead.

Question # 66

Date: February 02, 2024

A company stores petabytes of data in thousands of Amazon S3 buckets in the S3 Standard storage class. The data supports analytics workloads that have unpredictable and variable data access patterns.

The company does not access some data for months. However, the company must be able to retrieve all data within milliseconds. The company needs to optimize S3 storage costs.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost-optimized storage classes. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.
B. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequently. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.
C. Use S3 Intelligent-Tiering. Activate the Deep Archive Access tier.
D. Use S3 Intelligent-Tiering. Use the default access tier.

Correct Answer:

A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost-optimized storage classes. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.

B. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequently. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.

C. Use S3 Intelligent-Tiering. Activate the Deep Archive Access tier.

D. Use S3 Intelligent-Tiering. Use the default access tier.

Explanation:

The correct answer is D, which suggests using Amazon S3 Intelligent-Tiering with the default access tier. Let's break down why this is the best solution for the company's needs.

1. Unpredictable Access Patterns: The company has analytics workloads with unpredictable and variable data access patterns. This means that it's difficult to predict which data will be accessed frequently and which will not. S3 Intelligent-Tiering is specifically designed for such use cases. It automatically moves data between two access tiers—frequent and infrequent—based on changing access patterns without any operational overhead.

2. Quick Retrieval Requirement: The company must be able to retrieve all its data within milliseconds, even if some data has not been accessed for months. S3 Intelligent-Tiering provides millisecond retrieval times for objects in both its frequent and infrequent access tiers, which meets this requirement.

3. Cost Optimization: S3 Intelligent-Tiering helps optimize costs by moving objects that haven't been accessed for 30 consecutive days to the infrequent access tier, which is cheaper than the frequent access tier. This is done automatically, so it reduces the need for manual monitoring and management, thus lowering operational overhead.

4. Minimal Operational Overhead: The solution requires the least operational overhead because it automates the process of moving data between access tiers. There's no need to set up and manage complex lifecycle policies or continuously monitor access patterns, as would be required with some of the other options.

5. Default Access Tier: Using the default access tier of S3 Intelligent-Tiering is sufficient for scenarios where deep archival storage is not necessary, ensuring that data retrieval remains fast. It simplifies the setup by avoiding additional configurations like activating deep archive tiers, which are not needed for this scenario since millisecond retrieval is essential.

In summary, S3 Intelligent-Tiering with the default access tier provides a seamless, automated, and cost-effective way to manage data with unpredictable access patterns while ensuring quick access, meeting all the company's requirements with minimal effort.

Question # 67

Date: February 02, 2024

A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size.

A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.

Which solution will meet this requirement with the LEAST operational effort?

A. Create and run an Apache Spark job in an AWS Glue notebook. Configure the job to read the S3 file and calculate the number of distinct customers.
B. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file. Run SQL queries from Amazon Athena to calculate the number of distinct customers.
C. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.
D. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Correct Answer:

A. Create and run an Apache Spark job in an AWS Glue notebook. Configure the job to read the S3 file and calculate the number of distinct customers.

B. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file. Run SQL queries from Amazon Athena to calculate the number of distinct customers.

C. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.

D. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Explanation:

The correct answer is D: Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Here's why this solution is the best choice with the least operational effort:

1. AWS Glue DataBrew Overview: AWS Glue DataBrew is a data preparation tool that allows users to clean and normalize data directly within the AWS ecosystem. It provides a visual interface to simplify the process of preparing data for analysis, reducing the need for complex coding or data engineering expertise.

2. Ease of Use: DataBrew is designed to be user-friendly, allowing users to create data transformation recipes through a no-code or low-code interface. This significantly reduces the operational effort compared to writing custom code or managing complex data processing jobs.

3. Built-in Functions: DataBrew includes built-in functions, such as COUNT_DISTINCT, that can be used to perform common data aggregation tasks easily. This means you can quickly calculate the number of distinct customers by using these pre-defined functions, without having to manually script or configure data processing logic.

4. Direct Integration with S3: Since the customer data file is stored in Amazon S3, DataBrew can directly access and process this data without the need for additional data movement or ETL processes. This seamless integration simplifies the workflow and reduces setup time.

5. Minimal Infrastructure Management: DataBrew is a managed service, meaning AWS handles the underlying infrastructure, scaling, and resource management. This further reduces the operational overhead compared to other options that might require managing compute resources or cluster configurations.

Comparing to the other options:

- Option A (AWS Glue with Apache Spark): This involves writing and managing a Spark job, which requires more technical expertise and effort to set up and maintain. It is more complex than using DataBrew's straightforward interface.

- Option B (AWS Glue Crawler and Athena): While this option can work, it involves setting up a Glue Crawler and running Athena queries. This setup is more involved and requires additional configurations, which increases the operational effort compared to using DataBrew.

- Option C (Amazon EMR Serverless with Spark): Similar to Option A, this requires setting up and managing a Spark job, albeit in a serverless environment. It still necessitates knowledge of Spark and involves more operational steps than DataBrew.

In conclusion, AWS Glue DataBrew offers the least operational effort through its intuitive interface, built-in functions, and seamless integration with AWS services, making it the optimal choice for calculating the number of distinct customers in this scenario.

Question # 68

Date: March 07, 2024

A company receives .csv files that contain physical address data. The data is in columns that have the following names: Door_No, Street_Name, City, and Zip_Code. The company wants to create a single column to store these values in the following format:

[Image could not be loaded]

Which solution will meet this requirement with the LEAST coding effort?

A. Use AWS Glue DataBrew to read the files. Use the NEST_TO_ARRAY transformation to create the new column.
B. Use AWS Glue DataBrew to read the files. Use the NEST_TO_MAP transformation to create the new column.
C. Use AWS Glue DataBrew to read the files. Use the PIVOT transformation to create the new column.
D. Write a Lambda function in Python to read the files. Use the Python data dictionary type to create the new column.

Correct Answer:

A. Use AWS Glue DataBrew to read the files. Use the NEST_TO_ARRAY transformation to create the new column.

B. Use AWS Glue DataBrew to read the files. Use the NEST_TO_MAP transformation to create the new column.

C. Use AWS Glue DataBrew to read the files. Use the PIVOT transformation to create the new column.

D. Write a Lambda function in Python to read the files. Use the Python data dictionary type to create the new column.

Explanation:

NEST_TO_ARRAY would result in: [{"key": "key1", "value": "value1"}, {"key": "key2", "value": "value2"}, {"key": "key3", "value": "value3"}]

while NEST_TO_MAP results: { "key1": "value1", "key2": "value2", "key3": "value3" }
Therefore go with B

Question # 69

Date: February 02, 2024

A company receives call logs as Amazon S3 objects that contain sensitive customer information. The company must protect the S3 objects by using encryption. The company must also use encryption keys that only specific employees can access.

Which solution will meet these requirements with the LEAST effort?

A. Use an AWS CloudHSM cluster to store the encryption keys. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the objects. Deploy an IAM policy that restricts access to the CloudHSM cluster.
B. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer information. Restrict access to the keys that encrypt the objects.
C. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.
D. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Correct Answer:

A. Use an AWS CloudHSM cluster to store the encryption keys. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the objects. Deploy an IAM policy that restricts access to the CloudHSM cluster.

B. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer information. Restrict access to the keys that encrypt the objects.

C. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.

D. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Explanation:

The correct answer, option C, suggests using server-side encryption with AWS Key Management Service (KMS) keys (SSE-KMS) to encrypt the objects that contain customer information. Let's break down why this is the most suitable solution with the least effort:

1. Simplicity and Integration with AWS Services: - AWS Key Management Service (KMS) is fully integrated with Amazon S3, providing a seamless experience for encrypting and decrypting data. This integration means that the encryption and decryption process can be managed automatically by AWS, reducing the complexity for the user.

2. Granular Access Control: - With SSE-KMS, you can define and enforce fine-grained permissions using AWS Identity and Access Management (IAM) policies. This allows you to specify exactly which employees can use the KMS keys to encrypt and decrypt data. By configuring IAM policies, you ensure that only authorized employees have access to the encryption keys, thereby protecting sensitive data.

3. Ease of Management: - Managing encryption keys with AWS KMS is straightforward. It provides automated key rotation, centralized management, and monitoring, which simplifies the process of ensuring encryption keys are handled securely. This reduces the effort required compared to managing encryption keys manually or through other more complex systems.

4. Audit and Compliance: - AWS KMS provides audit logs that record every use of your encryption keys through AWS CloudTrail. This is crucial for compliance and security audits as it allows you to track who accessed the keys and when.

5. Cost-effective: - Using SSE-KMS is generally cost-effective compared to setting up a dedicated hardware security module (HSM) like CloudHSM. With AWS KMS, you don't have to worry about the underlying hardware, which reduces both setup and operational costs.

In contrast, the other options involve more complexity and effort: - Option A involves using AWS CloudHSM, which requires setting up and managing a hardware security appliance, a more complex and costly solution. - Option B requires handling customer-provided keys manually, which increases the risk of key management errors and adds complexity. - Option D uses Amazon S3 managed keys (SSE-S3), which do not provide the same level of control over access to the keys as SSE-KMS, making it unsuitable when specific access restrictions are needed.

Therefore, option C, using SSE-KMS, meets the company's requirements for encryption and access control with the least effort.

Question # 70

Date: February 02, 2024

A data engineer uses Amazon Redshift to run resource-intensive analytics processes once every month. Every month, the data engineer creates a new Redshift provisioned cluster. The data engineer deletes the Redshift provisioned cluster after the analytics processes are complete every month. Before the data engineer deletes the cluster each month, the data engineer unloads backup data from the cluster to an Amazon S3 bucket.

The data engineer needs a solution to run the monthly analytics processes that does not require the data engineer to manage the infrastructure manually.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month.
B. Use Amazon Redshift Serverless to automatically process the analytics workload.
C. Use the AWS CLI to automatically process the analytics workload.
D. Use AWS CloudFormation templates to automatically process the analytics workload.

Correct Answer:

A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month.

B. Use Amazon Redshift Serverless to automatically process the analytics workload.

C. Use the AWS CLI to automatically process the analytics workload.

D. Use AWS CloudFormation templates to automatically process the analytics workload.

Explanation:

Certainly! Let's break down the reasoning behind why option B, using Amazon Redshift Serverless, is the correct choice to meet the requirements with the least operational overhead.

The scenario involves a data engineer who is currently using Amazon Redshift to run monthly analytics processes. The engineer creates a new Redshift provisioned cluster every month and deletes it after the processes are complete, backing up data to Amazon S3 before deletion. The key requirement here is to automate this process and reduce manual management of the infrastructure, thus minimizing operational overhead.

Here's why Amazon Redshift Serverless is the ideal solution:

1. Serverless Model: Amazon Redshift Serverless allows you to run analytics workloads without having to manage the infrastructure. Unlike the traditional Redshift provisioned clusters, which require you to configure, scale, and manage the underlying hardware, Redshift Serverless abstracts these tasks away. This means you don't need to manually create or delete clusters every month.

2. Automatic Scaling: Redshift Serverless automatically provisions and scales resources based on your workload demands. This ensures that you have the right amount of compute power available when your monthly analytics processes run, and you are only charged for the actual usage, potentially reducing costs.

3. Ease of Use: Since it's serverless, you don't need to worry about starting up or shutting down clusters. This aligns perfectly with the requirement to reduce manual infrastructure management. You simply point your analytics processes to the serverless endpoint, and it takes care of the rest.

4. Operational Overhead: By choosing Redshift Serverless, the data engineer eliminates the need to manage cluster lifecycle, backups, and scaling. This greatly reduces operational overhead compared to managing a provisioned cluster, where you have to manually handle these tasks.

5. Cost Efficiency: With Redshift Serverless, you avoid paying for idle resources. The serverless model ensures that you are billed based on the actual usage during the analytics processing, rather than maintaining a running cluster that you only use once a month.

In contrast, the other options either involve some form of manual intervention or do not directly address the requirement to automate the infrastructure management:

- Option A involves pausing and resuming a provisioned cluster, which still requires manual or scripted intervention and management of the cluster. - Option C suggests using the AWS CLI, which still requires manual scripting and execution, thus not reducing operational overhead significantly. - Option D involves AWS CloudFormation, which automates the deployment of resources but still requires you to manage the lifecycle of a provisioned cluster.

Overall, Amazon Redshift Serverless is designed to simplify analytics workloads by removing the need for manual infrastructure management, making it the best choice for this scenario.

Question # 71

Date: June 15, 2024

An online retail company stores Application Load Balancer (ALB) access logs in an Amazon S3 bucket. The company wants to use Amazon Athena to query the logs to analyze traffic patterns.

A data engineer creates an unpartitioned table in Athena. As the amount of the data gradually increases, the response time for queries also increases. The data engineer wants to improve the query performance in Athena.

Which solution will meet these requirements with the LEAST operational effort?

A. Create an AWS Glue job that determines the schema of all ALB access logs and writes the partition metadata to AWS Glue Data Catalog.
B. Create an AWS Glue crawler that includes a classifier that determines the schema of all ALB access logs and writes the partition metadata to AWS Glue Data Catalog.
C. Create an AWS Lambda function to transform all ALB access logs. Save the results to Amazon S3 in Apache Parquet format. Partition the metadata. Use Athena to query the transformed data.
D. Use Apache Hive to create bucketed tables. Use an AWS Lambda function to transform all ALB access logs.

Correct Answer:

A. Create an AWS Glue job that determines the schema of all ALB access logs and writes the partition metadata to AWS Glue Data Catalog.

B. Create an AWS Glue crawler that includes a classifier that determines the schema of all ALB access logs and writes the partition metadata to AWS Glue Data Catalog.

C. Create an AWS Lambda function to transform all ALB access logs. Save the results to Amazon S3 in Apache Parquet format. Partition the metadata. Use Athena to query the transformed data.

D. Use Apache Hive to create bucketed tables. Use an AWS Lambda function to transform all ALB access logs.

Explanation:

The correct answer is B, which suggests creating an AWS Glue crawler with a classifier that determines the schema of all ALB access logs and writes the partition metadata to the AWS Glue Data Catalog. Let's break down why this solution is appropriate and efficient:

1. Understanding the Challenge: The online retail company is facing performance issues with Athena queries as the data volume grows. This is because the table in Athena is currently unpartitioned, meaning every query has to scan potentially large amounts of data, leading to increased response times.

2. The Role of Partitioning: Partitioning in Athena (and data lakes in general) helps improve query performance by reducing the amount of data scanned. By organizing data into partitions based on certain fields (like date, region, etc.), queries can be more selective about which parts of the data they need to scan. This drastically reduces the amount of data processed and speeds up query performance.

3. AWS Glue Crawler: An AWS Glue crawler is a tool that automatically detects the schema of datasets and can also determine how data should be partitioned. It writes this schema and partition metadata to the AWS Glue Data Catalog, which Athena then uses to understand how to query the data efficiently.

4. Least Operational Effort: The question specifically asks for a solution with the least operational effort. AWS Glue crawlers are designed to automate the discovery of data schemas and partitions. Once set up, they run on a schedule to continuously update the Glue Data Catalog as new data arrives, requiring minimal manual intervention and maintenance.

5. Why Other Options Are Less Suitable: - Option A: Involves manually setting up an AWS Glue job to determine schemas and write partition metadata. This requires more manual setup and maintenance compared to the automated nature of a Glue crawler. - Option C: Involves transforming logs with AWS Lambda and converting them to Parquet format, which is an efficient storage format but requires additional steps and infrastructure (setting up Lambda functions, transformation logic, etc.). This increases operational complexity. - Option D: Suggests using Apache Hive and Lambda for transformations, which adds additional layers of complexity and operational overhead compared to the automated Glue Crawler approach.

In summary, Option B with an AWS Glue crawler provides an automated, scalable solution that simplifies schema and partition management, enhancing query performance in Athena with minimal operational overhead.

Question # 72

Date: February 02, 2024

A data engineer needs to use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket. When the data engineer connects to the QuickSight dashboard, the data engineer receives an error message that indicates insufficient permissions.

Which factors could cause to the permissions-related errors? (Choose two.)

A. There is no connection between QuickSight and Athena.
B. The Athena tables are not cataloged.
C. QuickSight does not have access to the S3 bucket.
D. QuickSight does not have access to decrypt S3 data.
E. There is no IAM role assigned to QuickSight.

Correct Answer:

A. There is no connection between QuickSight and Athena.

B. The Athena tables are not cataloged.

C. QuickSight does not have access to the S3 bucket.

D. QuickSight does not have access to decrypt S3 data.

E. There is no IAM role assigned to QuickSight.

Explanation:

Certainly! Let's break down why options C and D are the correct answers to the question of why a data engineer might encounter permissions-related errors when trying to use an Amazon QuickSight dashboard based on Amazon Athena queries with data stored in an Amazon S3 bucket.

Option C: QuickSight does not have access to the S3 bucket.

Amazon QuickSight relies on access to data stored in Amazon S3 when creating visualizations based on Athena queries. Athena is a query service that allows you to run SQL queries on data stored in S3, and for QuickSight to visualize this data, it needs permission to access the underlying S3 bucket where the data resides. If QuickSight is not granted the necessary permissions to read from the S3 bucket, it will result in a permissions error. This is because QuickSight requires explicit permissions to list and read the objects in the S3 bucket to retrieve the query results from Athena. You can provide these permissions by ensuring that the IAM role or user associated with QuickSight has the appropriate Amazon S3 actions (like `s3:GetObject` and `s3:ListBucket`) allowed in its policy.

Option D: QuickSight does not have access to decrypt S3 data.

If the data in the S3 bucket is encrypted, QuickSight must have permissions to decrypt it in order to read and visualize the data. AWS supports various encryption methods for S3, such as using AWS Key Management Service (KMS) keys. If the data is encrypted using a KMS key, QuickSight needs permission to use the KMS key for decryption operations (like `kms:Decrypt`). Without these decryption permissions, even if QuickSight has access to the S3 bucket, it won't be able to retrieve the readable data, leading to permissions-related errors. Therefore, ensuring that QuickSight has the necessary KMS permissions is crucial when working with encrypted data.

In summary, both options C and D highlight essential access requirements for QuickSight to function correctly with Athena queries on S3-stored data. Without proper permissions to access the S3 bucket and decrypt the data, QuickSight will encounter errors, preventing the data engineer from successfully using the dashboard.

Question # 73

Date: February 04, 2024

A healthcare company uses Amazon Kinesis Data Streams to stream real-time health data from wearable devices, hospital equipment, and patient records.

A data engineer needs to find a solution to process the streaming data. The data engineer needs to store the data in an Amazon Redshift Serverless warehouse. The solution must support near real-time analytics of the streaming data and the previous day's data.

Which solution will meet these requirements with the LEAST operational overhead?

A. Load data into Amazon Kinesis Data Firehose. Load the data into Amazon Redshift.
B. Use the streaming ingestion feature of Amazon Redshift.
C. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.
D. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Correct Answer:

A. Load data into Amazon Kinesis Data Firehose. Load the data into Amazon Redshift.

B. Use the streaming ingestion feature of Amazon Redshift.

C. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.

D. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Explanation:

The correct answer is B: Use the streaming ingestion feature of Amazon Redshift.

Here's a detailed explanation of why this solution is the best fit with the least operational overhead:

1. Amazon Redshift Streaming Ingestion: This feature allows Amazon Redshift to natively ingest streaming data directly from Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose. It simplifies the process by allowing data to be brought into Redshift without the need for intermediate storage or complex ETL processes. This direct integration is designed to handle streaming data efficiently and is optimized for near real-time analytics.

2. Near Real-Time Analytics: The requirement is to support near real-time analytics of both streaming data and the previous day's data. With the streaming ingestion capability, Redshift can process and make streaming data available for analysis as soon as it arrives. This is crucial for applications that need up-to-date insights, such as monitoring health data from wearable devices.

3. Operational Overhead: Using Amazon Redshift's streaming ingestion minimizes operational overhead because it eliminates the need for additional data processing layers or services. There's no need to manage additional ETL processes or intermediate data storage solutions, such as Amazon S3, which are required in other options.

4. Scalability and Management: Since the question specifies using Amazon Redshift Serverless, this feature aligns well with the serverless model, which abstracts much of the infrastructure management. It automatically scales based on the data ingestion rate and analytics workload, further reducing operational complexity.

5. Alternative Options: - Option A involves using Amazon Kinesis Data Firehose to load data into Redshift. While this is a viable approach, it introduces an additional step and layer to manage, which increases operational overhead. - Option C suggests loading data into Amazon S3 first and then using the COPY command to load data into Redshift. This process adds latency and complexity, as it requires managing data in S3 and manually triggering or scheduling the COPY operation. - Option D involves using Amazon Aurora with zero-ETL integration, which is irrelevant for this use case since the data originates from Kinesis Data Streams and not from a relational database like Aurora.

In conclusion, Option B is the most efficient choice due to its direct integration, reduced complexity, and alignment with the requirement for near real-time data processing with minimal operational overhead.

Question # 74

Date: February 02, 2024

A company has multiple applications that use datasets that are stored in an Amazon S3 bucket. The company has an ecommerce application that generates a dataset that contains personally identifiable information (PII). The company has an internal analytics application that does not require access to the PII.

To comply with regulations, the company must not share PII unnecessarily. A data engineer needs to implement a solution that with redact PII dynamically, based on the needs of each application that accesses the dataset.

Which solution will meet the requirements with the LEAST operational overhead?

A. Create an S3 bucket policy to limit the access each application has. Create multiple copies of the dataset. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.
B. Create an S3 Object Lambda endpoint. Use the S3 Object Lambda endpoint to read data from the S3 bucket. Implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data.
C. Use AWS Glue to transform the data for each application. Create multiple copies of the dataset. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.
D. Create an API Gateway endpoint that has custom authorizers. Use the API Gateway endpoint to read data from the S3 bucket. Initiate a REST API call to dynamically redact PII based on the needs of each application that accesses the data.

Correct Answer:

A. Create an S3 bucket policy to limit the access each application has. Create multiple copies of the dataset. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.

B. Create an S3 Object Lambda endpoint. Use the S3 Object Lambda endpoint to read data from the S3 bucket. Implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data.

C. Use AWS Glue to transform the data for each application. Create multiple copies of the dataset. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.

D. Create an API Gateway endpoint that has custom authorizers. Use the API Gateway endpoint to read data from the S3 bucket. Initiate a REST API call to dynamically redact PII based on the needs of each application that accesses the data.

Explanation:

The correct answer is B: Create an S3 Object Lambda endpoint. Use the S3 Object Lambda endpoint to read data from the S3 bucket. Implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data.

Here's why this solution is the best fit with the least operational overhead:

1. Dynamic Redaction: S3 Object Lambda allows you to add your own code to process data retrieved from S3 before it is returned to the requesting application. By using S3 Object Lambda, you can implement logic that dynamically redacts PII based on the specific requirements of each application accessing the data. This means you can provide different views of the data to different applications without creating multiple copies.

2. Single Data Source: Unlike solutions that require maintaining multiple copies of the dataset (Options A and C), S3 Object Lambda enables you to use a single dataset stored in S3. This significantly reduces storage costs and operational complexity because you don't need to manage and synchronize multiple versions of the data.

3. Operational Overhead: With S3 Object Lambda, you minimize operational overhead since the redaction logic is encapsulated within a Lambda function. AWS Lambda is a fully managed service, which means you don't need to worry about provisioning or managing servers. The function will automatically scale to handle varying numbers of requests.

4. Security and Compliance: By dynamically redacting PII, you ensure compliance with data privacy regulations by sharing only the necessary data with each application. You can easily update the redaction logic as regulatory requirements change, without altering the dataset or creating new copies.

5. Integration: S3 Object Lambda integrates seamlessly with existing S3 operations, so applications can continue to use standard S3 APIs to access data. This makes implementation straightforward and reduces the need for significant application changes.

In comparison, the other options involve creating multiple copies of datasets (A and C) or setting up additional infrastructure like API Gateway with custom logic (D), both of which increase complexity and operational demands. S3 Object Lambda provides a streamlined and efficient solution for on-the-fly data processing, making it the optimal choice for this scenario.

Question # 75

Date: February 02, 2024

A data engineer needs to build an extract, transform, and load (ETL) job. The ETL job will process daily incoming .csv files that users upload to an Amazon S3 bucket. The size of each S3 object is less than 100 MB.

Which solution will meet these requirements MOST cost-effectively?

A. Write a custom Python application. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.

B. Write a PySpark ETL script. Host the script on an Amazon EMR cluster.
C. Write an AWS Glue PySpark job. Use Apache Spark to transform the data.
D. Write an AWS Glue Python shell job. Use pandas to transform the data.

Correct Answer:

A. Write a custom Python application. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.

B. Write a PySpark ETL script. Host the script on an Amazon EMR cluster.

C. Write an AWS Glue PySpark job. Use Apache Spark to transform the data.

D. Write an AWS Glue Python shell job. Use pandas to transform the data.

Explanation:

The task at hand is to build an ETL job for processing daily incoming .csv files that are uploaded to an Amazon S3 bucket, with each file being less than 100 MB. The requirement is to find the most cost-effective solution.

Let's break down the options:

A. Write a custom Python application. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster. - Amazon EKS is a managed Kubernetes service that enables you to run Kubernetes applications on AWS. While it's a robust platform, it is generally overkill for processing small .csv files of less than 100 MB. Setting up and maintaining an EKS cluster involves considerable overhead and costs, including the management of containerized applications, scaling, and paying for the underlying EC2 instances. This option doesn't align with the need for a cost-effective solution for small-scale ETL jobs.

B. Write a PySpark ETL script. Host the script on an Amazon EMR cluster. - Amazon EMR is a cloud big data platform for processing vast amounts of data using open-source tools such as Apache Spark and Hadoop. Although EMR can handle ETL tasks efficiently, it is designed for large-scale data processing. For small files like those described, the cost associated with running an EMR cluster might not be justified. EMR clusters incur costs based on the number and type of nodes used, which could be excessive for processing .csv files under 100 MB.

C. Write an AWS Glue PySpark job. Use Apache Spark to transform the data. - AWS Glue is a fully managed ETL service that makes it easy to prepare and load data. While Glue's PySpark jobs are suitable for complex transformations and large datasets, using PySpark may add unnecessary complexity and cost when dealing with smaller datasets. Glue's pricing is based on the number of Data Processing Units (DPUs) used, and for simple tasks on small datasets, this might not be the most economical choice.

D. Write an AWS Glue Python shell job. Use pandas to transform the data. - An AWS Glue Python shell job is a lightweight, serverless option for running Python scripts. It's a cost-effective choice for performing ETL operations on smaller datasets, such as .csv files under 100 MB. Using pandas, a Python data manipulation library, is efficient for handling small to medium data sizes and allows for straightforward data transformations without the overhead of setting up a larger processing cluster. AWS Glue Python shell jobs are billed based on compute time, making it a more economical option for less intensive processing tasks.

In summary, Option D is the most cost-effective solution because it combines the simplicity and efficiency of pandas for small data transformations with the serverless and lower-cost nature of AWS Glue Python shell jobs, avoiding the unnecessary complexity and cost associated with larger-scale processing solutions.

Question # 76

Date: June 14, 2024

A data engineer creates an AWS Glue Data Catalog table by using an AWS Glue crawler that is named Orders. The data engineer wants to add the following new partitions:

s3://transactions/orders/order_date=2023-01-01

s3://transactions/orders/order_date=2023-01-02

The data engineer must edit the metadata to include the new partitions in the table without scanning all the folders and files in the location of the table.

Which data definition language (DDL) statement should the data engineer use in Amazon Athena?

A. ALTER TABLE Orders ADD PARTITION(order_date='2023-01-01') LOCATION 's3://transactions/orders/order_date=2023-01-01';ALTER TABLE Orders ADD PARTITION(order_date='2023-01-02') LOCATION 's3://transactions/orders/order_date=2023-01-02';
B. MSCK REPAIR TABLE Orders;
C. REPAIR TABLE Orders;
D. ALTER TABLE Orders MODIFY PARTITION(order_date='2023-01-01') LOCATION 's3://transactions/orders/2023-01-01';ALTER TABLE Orders MODIFY PARTITION(order_date='2023-01-02') LOCATION 's3://transactions/orders/2023-01-02';

Correct Answer:

A. ALTER TABLE Orders ADD PARTITION(order_date='2023-01-01') LOCATION 's3://transactions/orders/order_date=2023-01-01';ALTER TABLE Orders ADD PARTITION(order_date='2023-01-02') LOCATION 's3://transactions/orders/order_date=2023-01-02';

B. MSCK REPAIR TABLE Orders;

C. REPAIR TABLE Orders;

D. ALTER TABLE Orders MODIFY PARTITION(order_date=’2023-01-01’) LOCATION ‘s3://transactions/orders/2023-01-01’;ALTER TABLE Orders MODIFY PARTITION(order_date=’2023-01-02’) LOCATION ‘s3://transactions/orders/2023-01-02’;

Explanation:

The correct answer is A, which involves using the `ALTER TABLE` statement in Amazon Athena to add partitions directly to the AWS Glue Data Catalog table.

Here's why this is the appropriate choice:

1. Understanding Partitions in AWS Glue and Athena: In AWS Glue and Athena, a table can be partitioned based on a specific column, such as `order_date` in this case. Partitions help organize the data better and can significantly improve query performance by reducing the amount of data that needs to be scanned.

2. Adding Partitions Manually: When a data engineer wants to add new partitions to an existing table without re-crawling the entire dataset, they can manually update the table's metadata to reflect these new partitions. This is essential when you have new data files that follow a specific partitioning scheme, like those under `s3://transactions/orders/order_date=2023-01-01`.

3. Using `ALTER TABLE ADD PARTITION`: The `ALTER TABLE ADD PARTITION` statement is specifically designed to add new partitions to a table's metadata without requiring a full scan of all the existing data files. This statement includes specifying the partition column value (e.g., `order_date=’2023-01-01’`) and the location of the data in S3.

4. Why Other Options Are Not Suitable: - Option B (`MSCK REPAIR TABLE Orders`): This command is used to automatically add partitions that exist in the S3 location but are not yet in the Glue Data Catalog. However, it involves scanning the entire location, which the data engineer wants to avoid. - Option C (`REPAIR TABLE Orders`): This is not a valid Athena command. Athena uses the `MSCK REPAIR TABLE` command to repair tables. - Option D (`ALTER TABLE MODIFY PARTITION`): This command is used to modify the location of an existing partition, not to add new partitions. It doesn’t serve the purpose of adding new partitions to the table.

In summary, the data engineer's requirement to add partitions without scanning all folders and files aligns perfectly with using the `ALTER TABLE ADD PARTITION` command, as it allows for precise and efficient updates to the table's metadata.

Question # 77

Date: June 14, 2024

A company stores 10 to 15 TB of uncompressed .csv files in Amazon S3. The company is evaluating Amazon Athena as a one-time query engine.

The company wants to transform the data to optimize query runtime and storage costs.

Which file format and compression solution will meet these requirements for Athena queries?

A. .csv format compressed with zip
B. JSON format compressed with bzip2
C. Apache Parquet format compressed with Snappy
D. Apache Avro format compressed with LZO

Correct Answer:

A. .csv format compressed with zip

B. JSON format compressed with bzip2

C. Apache Parquet format compressed with Snappy

D. Apache Avro format compressed with LZO

Explanation:

The correct answer is C: Apache Parquet format compressed with Snappy. Let's break down why this is the best choice for optimizing both query runtime and storage costs when using Amazon Athena.

1. File Format Efficiency: - Apache Parquet: Parquet is a columnar storage file format that is highly efficient for analytical queries. Instead of storing data row by row, it stores data column by column. This is advantageous for queries that only need to access a subset of columns, as it reduces the amount of data scanned, thus speeding up query performance and reducing costs associated with data scanning in Athena.

2. Compression: - Snappy Compression: Snappy is a compression algorithm that is designed to provide a good balance between compression speed and space efficiency. While it may not compress as tightly as some other algorithms, it is very fast, which is crucial for performance-sensitive applications like querying large datasets in Athena. The faster decompression time means quicker query execution.

3. Comparison with Other Options: - Option A (.csv format compressed with zip): While CSV is a widely used format, it is not optimized for columnar operations, which means Athena would have to scan more data than necessary. Additionally, ZIP compression is not as efficient in terms of speed for large datasets. - Option B (JSON format compressed with bzip2): JSON is a flexible format but is not designed for efficient querying of structured data, especially at the scale of 10 to 15 TB. Bzip2 offers high compression but is slower in both compression and decompression, which could negatively impact query performance. - Option D (Apache Avro format compressed with LZO): Avro is another data serialization system, but it is typically row-oriented, which is less efficient than columnar formats like Parquet for analytical queries. LZO provides fast compression and decompression, but the combination of Avro and LZO does not offer the same query performance benefits as Parquet with Snappy.

In summary, Apache Parquet combined with Snappy compression is the strongest choice for optimizing both query runtime and storage costs in Amazon Athena. This setup leverages the strengths of columnar storage and efficient compression to handle large-scale data efficiently.

Question # 78

Date: June 15, 2024

A company uses Apache Airflow to orchestrate the company's current on-premises data pipelines. The company runs SQL data quality check tasks as part of the pipelines. The company wants to migrate the pipelines to AWS and to use AWS managed services.

Which solution will meet these requirements with the LEAST amount of refactoring?

A. Setup AWS Outposts in the AWS Region that is nearest to the location where the company uses Airflow. Migrate the servers into Outposts hosted Amazon EC2 instances. Update the pipelines to interact with the Outposts hosted EC2 instances instead of the on-premises pipelines.
B. Create a custom Amazon Machine Image (AMI) that contains the Airflow application and the code that the company needs to migrate. Use the custom AMI to deploy Amazon EC2 instances. Update the network connections to interact with the newly deployed EC2 instances.
C. Migrate the existing Airflow orchestration configuration into Amazon Managed Workflows for Apache Airflow (Amazon MWAA). Create the data quality checks during the ingestion to validate the data quality by using SQL tasks in Airflow.
D. Convert the pipelines to AWS Step Functions workflows. Recreate the data quality checks in SQL as Python based AWS Lambda functions.

Correct Answer:

A. Setup AWS Outposts in the AWS Region that is nearest to the location where the company uses Airflow. Migrate the servers into Outposts hosted Amazon EC2 instances. Update the pipelines to interact with the Outposts hosted EC2 instances instead of the on-premises pipelines.

B. Create a custom Amazon Machine Image (AMI) that contains the Airflow application and the code that the company needs to migrate. Use the custom AMI to deploy Amazon EC2 instances. Update the network connections to interact with the newly deployed EC2 instances.

C. Migrate the existing Airflow orchestration configuration into Amazon Managed Workflows for Apache Airflow (Amazon MWAA). Create the data quality checks during the ingestion to validate the data quality by using SQL tasks in Airflow.

D. Convert the pipelines to AWS Step Functions workflows. Recreate the data quality checks in SQL as Python based AWS Lambda functions.

Explanation:

The correct answer is C: Migrate the existing Airflow orchestration configuration into Amazon Managed Workflows for Apache Airflow (Amazon MWAA). Create the data quality checks during the ingestion to validate the data quality by using SQL tasks in Airflow.

Here's why this solution is ideal with the least amount of refactoring:

1. Amazon Managed Workflows for Apache Airflow (MWAA): This AWS service provides a managed environment for running Apache Airflow, which means you can continue using the same orchestration tool (Airflow) that your company is already familiar with. This greatly reduces the need for extensive changes to your workflows and minimizes the learning curve for your team.

2. Minimal Changes Required: By migrating to Amazon MWAA, you essentially lift and shift your existing configurations and DAGs (Directed Acyclic Graphs) from your on-premises setup to AWS. This approach allows you to retain your existing data pipeline logic and structure, avoiding the substantial refactoring that would be required if you were to switch to a different orchestration tool or workflow service.

3. Integration with Existing SQL Tasks: Since your current pipelines include SQL data quality checks, using MWAA allows you to continue running these checks within your Airflow tasks. You can seamlessly integrate these SQL tasks into the MWAA environment, ensuring that your data validation processes remain intact and functional.

4. AWS Managed Service Benefits: By using a managed service like MWAA, you offload the operational overhead of maintaining Airflow infrastructure, such as scaling, patching, and availability. This allows your team to focus more on developing and optimizing the data pipelines rather than managing the underlying architecture.

5. Cost Efficiency and Resource Optimization: With MWAA, AWS takes care of infrastructure management, which can lead to cost savings and resource optimization. You pay for what you use, and AWS handles the scaling, which can be more cost-effective than managing your own Airflow environment on EC2 instances.

In contrast, the other options would involve significant changes to your current setup. For example, setting up AWS Outposts (Option A) involves deploying physical hardware, which is costly and complex. Creating custom AMIs and deploying EC2 instances (Option B) requires more manual management and doesn't leverage the benefits of a managed service. Finally, converting pipelines to AWS Step Functions (Option D) and recreating SQL checks as Lambda functions would require a complete rewrite of your workflows, which is not only time-consuming but also increases the risk of introducing errors.

Overall, Option C provides a smooth transition to AWS with minimal disruption to your existing workflows, leveraging a managed service that aligns closely with your current setup.

Question # 79

Date: June 14, 2024

A company uses Amazon EMR as an extract, transform, and load (ETL) pipeline to transform data that comes from multiple sources. A data engineer must orchestrate the pipeline to maximize performance.

Which AWS service will meet this requirement MOST cost effectively?

A. Amazon EventBridge
B. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
C. AWS Step Functions
D. AWS Glue Workflows

Correct Answer:

A. Amazon EventBridge

B. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

C. AWS Step Functions

D. AWS Glue Workflows

Explanation:

AWS Step Functions is the correct answer for orchestrating an ETL pipeline involving Amazon EMR to maximize performance in a cost-effective manner. Here's why:

1. AWS Step Functions Overview: AWS Step Functions is a serverless orchestration service that allows you to coordinate multiple AWS services into serverless workflows. It enables you to define state machines using JSON-based Amazon States Language (ASL) to control the flow of your application and manage its execution.

2. Integration with Amazon EMR: Step Functions integrates seamlessly with Amazon EMR. You can initiate, track, and manage EMR cluster operations, such as starting a cluster, submitting steps to a cluster, and shutting it down, all within a Step Functions workflow. This integration simplifies the management of complex job sequences and dependencies on EMR.

3. Cost-Effectiveness: Step Functions charges based on the number of state transitions and the duration of each execution, making it a cost-effective solution for orchestrating complex workflows. Unlike some other services, it does not require upfront infrastructure costs or long-term commitments, allowing you to pay only for what you use.

4. Simplicity and Visibility: Step Functions provides a visual workflow design, making it easier to design, test, and audit the execution of your ETL pipeline. You can monitor the entire workflow execution in a single view, which aids in debugging and optimizing performance.

5. Reliability and Error Handling: Step Functions offers built-in error handling, automatic retries, and the ability to define fallback strategies. This reliability ensures your ETL processes are robust and can handle failures gracefully, improving the overall performance and consistency of the pipeline.

6. Alternatives Considered: - Amazon EventBridge (Option A) is more suited for event-driven architectures and does not provide the orchestration capabilities needed for an ETL pipeline. - Amazon Managed Workflows for Apache Airflow (Amazon MWAA) (Option B) is another orchestration tool but can be more complex and expensive due to its need for infrastructure setup and management. - AWS Glue Workflows (Option D) is specifically designed for AWS Glue jobs and might not integrate as well with EMR compared to Step Functions, which offers a broader orchestration capability.

In conclusion, AWS Step Functions is the most cost-effective and efficient solution for orchestrating an ETL pipeline with Amazon EMR, providing seamless integration, flexible error handling, and a pay-as-you-go pricing model.

Question # 80

Date: August 14, 2024

A data engineer has implemented data quality rules in 1,000 AWS Glue Data Catalog tables. Because of a recent change in business requirements, the data engineer must edit the data quality rules.

How should the data engineer meet this requirement with the LEAST operational overhead?

A. Create a pipeline in AWS Glue ETL to edit the rules for each of the 1,000 Data Catalog tables. Use an AWS Lambda function to call the corresponding AWS Glue job for each Data Catalog table.
B. Create an AWS Lambda function that makes an API call to AWS Glue Data Quality to make the edits.
C. Create an Amazon EMR cluster. Run a pipeline on Amazon EMR that edits the rules for each Data Catalog table. Use an AWS Lambda function to run the EMR pipeline.
D. Use the AWS Management Console to edit the rules within the Data Catalog.

Correct Answer:

A. Create a pipeline in AWS Glue ETL to edit the rules for each of the 1,000 Data Catalog tables. Use an AWS Lambda function to call the corresponding AWS Glue job for each Data Catalog table.

B. Create an AWS Lambda function that makes an API call to AWS Glue Data Quality to make the edits.

C. Create an Amazon EMR cluster. Run a pipeline on Amazon EMR that edits the rules for each Data Catalog table. Use an AWS Lambda function to run the EMR pipeline.

D. Use the AWS Management Console to edit the rules within the Data Catalog.

Explanation:

The correct answer, option B, suggests creating an AWS Lambda function that makes an API call to AWS Glue Data Quality to make the edits. Let's break down why this is the most efficient solution with the least operational overhead:

1. Automation and Scalability: By leveraging AWS Lambda, you can automate the process of editing data quality rules across all 1,000 tables. Lambda functions can be triggered programmatically, which allows you to avoid manual interventions. This approach is inherently scalable as Lambda can handle concurrent executions, making it suitable for processing a large number of tables efficiently.

2. API Integration: AWS Glue Data Quality provides APIs that allow you to programmatically interact with data quality rules. By using these APIs, you can directly edit the rules without having to process data manually or create complex workflows. This reduces both the time needed to implement changes and the risk of human error.

3. Minimal Operational Overhead: Managing a Lambda function involves minimal operational overhead compared to setting up and maintaining an ETL pipeline or an EMR cluster. Lambda is fully managed by AWS, which means you don't have to worry about infrastructure management, scalability, or fault tolerance.

4. Cost-Effectiveness: AWS Lambda follows a pay-as-you-go pricing model, which means you only pay for the compute time you consume. This can be more cost-effective than running an Amazon EMR cluster, which incurs costs for the compute resources it uses, even if they are underutilized.

5. Flexibility and Speed: Editing rules using Lambda and API calls can be quick and flexible, allowing you to apply changes rapidly across all tables. This is particularly beneficial when dealing with frequent changes in business requirements, as it allows the data engineer to react promptly without extensive reconfiguration.

In contrast, the other options either involve more manual steps (option D), require maintaining additional infrastructure (option C), or involve creating a more complex ETL pipeline (option A), which all add operational complexity and overhead. Therefore, option B is the most streamlined and efficient approach for editing data quality rules in this scenario.

Question # 81

Date: June 15, 2024

A company has a data lake on AWS. The data lake ingests sources of data from business units. The company uses Amazon Athena for queries. The storage layer is Amazon S3 with an AWS Glue Data Catalog as a metadata repository.

The company wants to make the data available to data scientists and business analysts. However, the company first needs to manage fine-grained, column-level data access for Athena based on the user roles and responsibilities.

Which solution will meet these requirements?

A. Set up AWS Lake Formation. Define security policy-based rules for the users and applications by IAM role in Lake Formation.
B. Define an IAM resource-based policy for AWS Glue tables. Attach the same policy to IAM user groups.
C. Define an IAM identity-based policy for AWS Glue tables. Attach the same policy to IAM roles. Associate the IAM roles with IAM groups that contain the users.
D. Create a resource share in AWS Resource Access Manager (AWS RAM) to grant access to IAM users.

Correct Answer:

A. Set up AWS Lake Formation. Define security policy-based rules for the users and applications by IAM role in Lake Formation.

B. Define an IAM resource-based policy for AWS Glue tables. Attach the same policy to IAM user groups.

C. Define an IAM identity-based policy for AWS Glue tables. Attach the same policy to IAM roles. Associate the IAM roles with IAM groups that contain the users.

D. Create a resource share in AWS Resource Access Manager (AWS RAM) to grant access to IAM users.

Explanation:

The correct answer is A: Set up AWS Lake Formation. Define security policy-based rules for the users and applications by IAM role in Lake Formation.

Here's why this is the best solution:

1. AWS Lake Formation Overview: AWS Lake Formation is a service designed to simplify the process of setting up a secure data lake on AWS. It allows you to manage access to data stored in Amazon S3 using fine-grained access controls. These controls include column-level permissions, which are essential for managing access based on user roles and responsibilities.

2. Fine-Grained Access Control: The requirement is to manage fine-grained, column-level data access for Amazon Athena. Lake Formation provides the ability to define access permissions at both the table and column levels. This means you can restrict access to specific columns in a table, ensuring that users only have access to the data they are authorized to see.

3. Integration with IAM: Lake Formation integrates with AWS Identity and Access Management (IAM), allowing you to define security policies based on IAM roles. This integration enables you to apply access controls that are consistent with your organization's security policies and user roles.

4. User Roles and Responsibilities: By defining security policy-based rules in Lake Formation, you can tailor access permissions to align with the specific roles and responsibilities of data scientists and business analysts. This ensures that data access is both secure and appropriate for the tasks each user needs to perform.

5. Centralized Access Management: Lake Formation provides a centralized platform to manage access to data across multiple services, such as Amazon Athena and AWS Glue. This centralization simplifies the management of permissions and ensures consistency in access controls.

The other options do not meet the requirements as effectively as AWS Lake Formation:

- Option B (IAM resource-based policy for AWS Glue tables) and Option C (IAM identity-based policy for AWS Glue tables) focus on AWS Glue tables but lack the column-level granularity and centralized access management that Lake Formation offers.

- Option D (AWS Resource Access Manager) is not suitable for this use case as it is primarily used for sharing resources across AWS accounts, not for managing fine-grained access to data within a data lake.

Therefore, setting up AWS Lake Formation and defining security policy-based rules for users and applications by IAM role is the most appropriate and effective solution for managing fine-grained, column-level access in this scenario.

Question # 82

Date: June 14, 2024

A retail company uses Amazon Aurora PostgreSQL to process and store live transactional data. The company uses an Amazon Redshift cluster for a data warehouse.

An extract, transform, and load (ETL) job runs every morning to update the Redshift cluster with new data from the PostgreSQL database. The company has grown rapidly and needs to cost optimize the Redshift cluster.

A data engineer needs to create a solution to archive historical data. The data engineer must be able to run analytics queries that effectively combine data from live transactional data in PostgreSQL, current data in Redshift, and archived historical data. The solution must keep only the most recent 15 months of data in Amazon Redshift to reduce costs.

Which combination of steps will meet these requirements? (Choose two.)

A. Configure the Amazon Redshift Federated Query feature to query live transactional data that is in the PostgreSQL database.
B. Configure Amazon Redshift Spectrum to query live transactional data that is in the PostgreSQL database.
C. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Amazon Redshift Spectrum to access historical data in Amazon S3.
D. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 Glacier Flexible Retrieval by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Redshift Spectrum to access historical data from S3 Glacier Flexible Retrieval.
E. Create a materialized view in Amazon Redshift that combines live, current, and historical data from different sources.

Correct Answer:

A. Configure the Amazon Redshift Federated Query feature to query live transactional data that is in the PostgreSQL database.

B. Configure Amazon Redshift Spectrum to query live transactional data that is in the PostgreSQL database.

C. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Amazon Redshift Spectrum to access historical data in Amazon S3.

D. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 Glacier Flexible Retrieval by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Redshift Spectrum to access historical data from S3 Glacier Flexible Retrieval.

E. Create a materialized view in Amazon Redshift that combines live, current, and historical data from different sources.

Explanation:

The correct answer is a combination of steps A and C. Let's break down why these steps are suitable for the given scenario.

First, let's consider the requirements:

1. Cost Optimization for Amazon Redshift: The company wants to reduce costs by keeping only the most recent 15 months of data in Amazon Redshift.

2. Ability to Query Across Different Data Sources: The solution must allow querying across the live transactional data in Amazon Aurora PostgreSQL, the current data in Amazon Redshift, and archived historical data.

3. Archiving Historical Data: The historical data should be archived after 15 months.

Now, let's look at each component:

A. Configure the Amazon Redshift Federated Query feature to query live transactional data that is in the PostgreSQL database.

- Why this step is correct: Amazon Redshift Federated Query allows you to query data across your Amazon Redshift data warehouse and your PostgreSQL database without moving the data. This feature enables real-time access to live transactional data stored in PostgreSQL, thus meeting the requirement to combine live data with current and historical data in queries. It is an efficient way to handle live data without duplicating it into Redshift, which helps in cost optimization as well.

C. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Amazon Redshift Spectrum to access historical data in Amazon S3.

- Why this step is correct: This step addresses the need to archive data older than 15 months to reduce storage costs in Redshift. By using the UNLOAD command, data is moved to Amazon S3, a cheaper storage option. Redshift Spectrum then allows Redshift to query this data directly from S3 without having to load it back into Redshift. This setup ensures that historical data remains accessible for queries while keeping Redshift costs optimized by only storing necessary data.

Taken together, these two steps provide a comprehensive solution that meets all the requirements. The use of Redshift Federated Query allows seamless querying across live and current data, while moving historical data to S3 and using Redshift Spectrum allows for cost-effective storage and querying of archived data. This combination effectively balances the need for cost optimization with the ability to perform complex analytics queries across different data sets.

Question # 83

Date: June 15, 2024

A manufacturing company has many IoT devices in facilities around the world. The company uses Amazon Kinesis Data Streams to collect data from the devices. The data includes device ID, capture date, measurement type, measurement value, and facility ID. The company uses facility ID as the partition key.

The company's operations team recently observed many WriteThroughputExceeded exceptions. The operations team found that some shards were heavily used but other shards were generally idle.

How should the company resolve the issues that the operations team observed?

A. Change the partition key from facility ID to a randomly generated key.
B. Increase the number of shards.
C. Archive the data on the producer's side.
D. Change the partition key from facility ID to capture date.

Correct Answer:

A. Change the partition key from facility ID to a randomly generated key.

B. Increase the number of shards.

C. Archive the data on the producer's side.

D. Change the partition key from facility ID to capture date.

Explanation:

In this situation, the company is experiencing "WriteThroughputExceeded" exceptions with Amazon Kinesis Data Streams, which indicates that the data being sent to a particular shard is exceeding the shard's capacity. This is often a result of uneven data distribution across shards, leading to some shards being overloaded while others remain underutilized.

The key to resolving this issue lies in the partition key. The partition key is used by Kinesis Data Streams to determine which shard a given record is sent to. In this case, the company is using the "facility ID" as the partition key. If certain facilities generate more data than others or if there are a few facilities with a large volume of devices, this can lead to some shards receiving a disproportionate amount of data, hence the "WriteThroughputExceeded" exceptions.

Let's go through the options:

A. Change the partition key from facility ID to a randomly generated key. - This option is correct because using a random key for partitioning would help distribute the data more evenly across all shards. Since each record would be assigned to a shard based on a random key, the likelihood of any single shard being overwhelmed is reduced. This leads to a more balanced load across the data stream, helping to avoid the throughput exceptions.

B. Increase the number of shards. - While increasing the number of shards could help handle more throughput overall, it doesn't address the core issue of uneven data distribution. Without changing the partitioning strategy, the same shards might still become overloaded, and the problem could persist.

C. Archive the data on the producer's side. - Archiving data on the producer's side doesn't solve the issue of throughput exceedance in real-time data streams. It may reduce the amount of data being sent temporarily, but it doesn't ensure a balanced load across shards, which is the root cause of the problem.

D. Change the partition key from facility ID to capture date. - Changing the partition key to capture date is unlikely to solve the problem because capture dates can be repetitive or clustered in ways that still lead to uneven distribution. For example, devices might send data at the same time intervals, leading to bursts that could overwhelm specific shards.

In summary, the best solution to resolve the WriteThroughputExceeded exceptions due to uneven shard usage is to alter the partition key to something that evenly distributes data across all shards. A randomly generated key is effective in achieving this balanced distribution, which is why option A is the correct answer.

Question # 84

Date: June 15, 2024

A data engineer wants to improve the performance of SQL queries in Amazon Athena that run against a sales data table.

The data engineer wants to understand the execution plan of a specific SQL statement. The data engineer also wants to see the computational cost of each operation in a SQL query.

Which statement does the data engineer need to run to meet these requirements?

A. EXPLAIN SELECT * FROM sales;
B. EXPLAIN ANALYZE FROM sales;
C. EXPLAIN ANALYZE SELECT * FROM sales;
D. EXPLAIN FROM sales;

Correct Answer:

A. EXPLAIN SELECT * FROM sales;

B. EXPLAIN ANALYZE FROM sales;

C. EXPLAIN ANALYZE SELECT * FROM sales;

D. EXPLAIN FROM sales;

Explanation:

To understand why option C, "EXPLAIN ANALYZE SELECT FROM sales;", is the correct answer, let's break down the purpose and functionality of the EXPLAIN and EXPLAIN ANALYZE statements in Amazon Athena.

1. EXPLAIN Statement: - The EXPLAIN statement in SQL is used to show the execution plan of a query. It provides details on how the query will be executed, such as the sequence of operations, join methods, and order of data access. This helps in understanding the query execution path but does not provide runtime statistics or computational costs.

2. EXPLAIN ANALYZE Statement: - The EXPLAIN ANALYZE statement extends the functionality of EXPLAIN by not only showing the execution plan but also executing the query and providing runtime statistics. This includes computational costs for each operation in the query. It offers insights into how much time each step of the query takes and how much data is processed at each step, which is crucial for performance tuning.

Given these definitions:

- Option A, "EXPLAIN SELECT FROM sales;", would only provide the execution plan without runtime statistics or computational costs. - Option B, "EXPLAIN ANALYZE FROM sales;", is syntactically incorrect because it doesn't follow the proper format of EXPLAIN ANALYZE, which should be followed by a full SQL SELECT statement. - Option D, "EXPLAIN FROM sales;", is also incorrect because it is missing the SELECT statement and does not adhere to correct SQL syntax.

Therefore, option C, "EXPLAIN ANALYZE SELECT FROM sales;", is the correct choice because it provides both the execution plan and the computational cost of each operation in the query. This allows the data engineer to understand the performance implications of their SQL statement fully, meeting the requirements stated in the question.

Question # 85

Date: June 15, 2024

A company plans to provision a log delivery stream within a VPC. The company configured the VPC flow logs to publish to Amazon CloudWatch Logs. The company needs to send the flow logs to Splunk in near real time for further analysis.

Which solution will meet these requirements with the LEAST operational overhead?

A. Configure an Amazon Kinesis Data Streams data stream to use Splunk as the destination. Create a CloudWatch Logs subscription filter to send log events to the data stream.
B. Create an Amazon Kinesis Data Firehose delivery stream to use Splunk as the destination. Create a CloudWatch Logs subscription filter to send log events to the delivery stream.
C. Create an Amazon Kinesis Data Firehose delivery stream to use Splunk as the destination. Create an AWS Lambda function to send the flow logs from CloudWatch Logs to the delivery stream.

D. Configure an Amazon Kinesis Data Streams data stream to use Splunk as the destination. Create an AWS Lambda function to send the flow logs from CloudWatch Logs to the data stream.

Correct Answer:

A. Configure an Amazon Kinesis Data Streams data stream to use Splunk as the destination. Create a CloudWatch Logs subscription filter to send log events to the data stream.

B. Create an Amazon Kinesis Data Firehose delivery stream to use Splunk as the destination. Create a CloudWatch Logs subscription filter to send log events to the delivery stream.

C. Create an Amazon Kinesis Data Firehose delivery stream to use Splunk as the destination. Create an AWS Lambda function to send the flow logs from CloudWatch Logs to the delivery stream.

D. Configure an Amazon Kinesis Data Streams data stream to use Splunk as the destination. Create an AWS Lambda function to send the flow logs from CloudWatch Logs to the data stream.

Explanation:

The correct answer is B, which involves using Amazon Kinesis Data Firehose to send VPC flow logs from Amazon CloudWatch Logs to Splunk. Let's break down why this is the optimal solution with the least operational overhead:

1. Amazon Kinesis Data Firehose: This service is specifically designed for loading streaming data into data stores and analytics services. It is fully managed, meaning that it automatically scales to adjust to the data throughput, which minimizes the need for operational management. When configured to use Splunk as the destination, Kinesis Data Firehose can directly deliver data to Splunk without requiring additional infrastructure.

2. CloudWatch Logs Subscription Filter: By creating a subscription filter, you can direct specific log events from CloudWatch Logs to flow into the Kinesis Data Firehose delivery stream. This setup allows for near real-time streaming of log data. Subscription filters are straightforward to configure and do not require additional compute resources or complex management.

3. Minimal Operational Overhead: Option B minimizes operational overhead by leveraging the native integration between CloudWatch Logs, Kinesis Data Firehose, and Splunk. This solution does not require you to manage or maintain additional infrastructure like an AWS Lambda function, which would be the case with options C and D. Additionally, Kinesis Data Streams (mentioned in options A and D) would require more manual management of scaling and partitioning, increasing the overhead.

4. Direct Integration with Splunk: Kinesis Data Firehose provides built-in support to deliver data to Splunk, including options for data transformation and retry logic. This direct integration simplifies the pipeline and ensures reliable data delivery without the need for custom coding or additional processing layers.

In summary, option B is the correct choice because it leverages a managed service that natively integrates with both the source (CloudWatch Logs) and the destination (Splunk), thus offering a seamless and efficient solution with minimal operational complexity.

Question # 86

Date: June 29, 2024

A banking company uses an application to collect large volumes of transactional data. The company uses Amazon Kinesis Data Streams for real-time analytics. The company's application uses the PutRecord action to send data to Kinesis Data Streams.

A data engineer has observed network outages during certain times of day. The data engineer wants to configure exactly-once delivery for the entire processing pipeline.

Which solution will meet this requirement?

A. Design the application so it can remove duplicates during processing by embedding a unique ID in each record at the source.
B. Update the checkpoint configuration of the Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) data collection application to avoid duplicate processing of events.
C. Design the data source so events are not ingested into Kinesis Data Streams multiple times.
D. Stop using Kinesis Data Streams. Use Amazon EMR instead. Use Apache Flink and Apache Spark Streaming in Amazon EMR.

Correct Answer:

A. Design the application so it can remove duplicates during processing by embedding a unique ID in each record at the source.

B. Update the checkpoint configuration of the Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) data collection application to avoid duplicate processing of events.

C. Design the data source so events are not ingested into Kinesis Data Streams multiple times.

D. Stop using Kinesis Data Streams. Use Amazon EMR instead. Use Apache Flink and Apache Spark Streaming in Amazon EMR.

Explanation:

The correct answer is A, which involves designing the application to handle deduplication by embedding a unique ID in each record at the source. Let's break down why this is the most appropriate solution for ensuring exactly-once delivery in this scenario:

1. Understanding Exactly-Once Delivery: Exactly-once delivery means that each record is processed once and only once, which is crucial for maintaining data integrity and consistency, especially in financial transactions where duplicate records can lead to significant errors.

2. Characteristics of Kinesis Data Streams: Amazon Kinesis Data Streams is designed for real-time data streaming, allowing applications to capture and process large volumes of data. While Kinesis ensures that data is delivered at least once, it doesn't inherently provide exactly-once delivery. This means that during network outages or retries, it's possible for records to be delivered more than once.

3. Role of Unique IDs: By embedding a unique ID in each record at the source, the application can distinguish between new and duplicate records. This is a common pattern for achieving idempotency, where processing the same record multiple times does not change the outcome beyond the initial application. Thus, the application can effectively remove duplicates, ensuring that each piece of data is processed exactly once.

4. Why Other Options Are Less Suitable: - Option B: Updating the checkpoint configuration in Amazon Managed Service for Apache Flink only helps with managing state and processing within Flink applications. It doesn't address deduplication at the source or handle network outages that might cause duplicate data to be sent initially. - Option C: Designing the data source to prevent multiple ingestions is ideal but not always feasible, especially in environments with intermittent network connectivity, where retries might occur. - Option D: Switching to Amazon EMR with Apache Flink and Apache Spark Streaming involves a complete overhaul of the architecture, which is unnecessarily complex for achieving exactly-once semantics when the issue can be resolved at the application level with unique IDs.

In summary, Option A is the most practical and direct solution to ensure exactly-once delivery by embedding a unique ID in each record, allowing the application to identify and handle duplicates effectively. This approach aligns with best practices in data processing and is well-suited to the capabilities of Kinesis Data Streams.

Question # 87

Date: June 15, 2024

A company has developed several AWS Glue extract, transform, and load (ETL) jobs to validate and transform data from Amazon S3. The ETL jobs load the data into Amazon RDS for MySQL in batches once every day. The ETL jobs use a DynamicFrame to read the S3 data.

The ETL jobs currently process all the data that is in the S3 bucket. However, the company wants the jobs to process only the daily incremental data.

Which solution will meet this requirement with the LEAST coding effort?

A. Create an ETL job that reads the S3 file status and logs the status in Amazon DynamoDB.
B. Enable job bookmarks for the ETL jobs to update the state after a run to keep track of previously processed data.
C. Enable job metrics for the ETL jobs to help keep track of processed objects in Amazon CloudWatch.
D. Configure the ETL jobs to delete processed objects from Amazon S3 after each run.

Correct Answer:

A. Create an ETL job that reads the S3 file status and logs the status in Amazon DynamoDB.

B. Enable job bookmarks for the ETL jobs to update the state after a run to keep track of previously processed data.

C. Enable job metrics for the ETL jobs to help keep track of processed objects in Amazon CloudWatch.

D. Configure the ETL jobs to delete processed objects from Amazon S3 after each run.

Explanation:

The correct answer is B: Enable job bookmarks for the ETL jobs to update the state after a run to keep track of previously processed data.

Here's why this is the most suitable solution with the least coding effort:

1. Understanding AWS Glue Job Bookmarks: AWS Glue job bookmarks are designed to help manage state information for ETL jobs. They allow AWS Glue to keep track of what data has been processed in previous runs. This feature is particularly useful for handling incremental data processing, where only new or changed data needs to be processed in subsequent job runs.

2. How Job Bookmarks Work: When job bookmarks are enabled, AWS Glue automatically tracks the progress of data processing for each job run. It records the state of processed data so that subsequent runs of the ETL job only process new or modified data. This eliminates the need to manually track which files or records have already been processed, thereby reducing coding effort.

3. Minimal Coding Effort: Enabling job bookmarks is a configuration change rather than a coding task. You simply need to enable the bookmarks feature in the AWS Glue ETL job settings. This requires no additional coding to keep track of processed data, as AWS Glue handles this automatically.

4. Scalability and Efficiency: Using job bookmarks is a scalable solution that integrates seamlessly with AWS Glue’s existing functionality. It efficiently manages large datasets by avoiding reprocessing of data, thus saving on processing time and costs.

5. Comparing with Other Options: - Option A: Creating an ETL job to log file statuses in DynamoDB would require additional development to implement and maintain the logic for tracking processed files. This involves more coding and complexity compared to simply enabling job bookmarks. - Option C: Using job metrics in CloudWatch is more about monitoring and alerting on job performance rather than tracking processed data. It doesn't inherently solve the problem of processing only new data. - Option D: Deleting processed objects from S3 is a more destructive approach and may not be suitable if data needs to be retained for compliance or audit purposes. It also doesn't inherently solve the problem of tracking incremental changes without additional logic.

In summary, enabling job bookmarks is the most efficient and low-effort method to achieve incremental data processing in AWS Glue, as it leverages built-in AWS capabilities to automatically track and manage processed data.

Question # 88

Date: June 15, 2024

An online retail company has an application that runs on Amazon EC2 instances that are in a VPC. The company wants to collect flow logs for the VPC and analyze network traffic.

Which solution will meet these requirements MOST cost-effectively?

A. Publish flow logs to Amazon CloudWatch Logs. Use Amazon Athena for analytics.
B. Publish flow logs to Amazon CloudWatch Logs. Use an Amazon OpenSearch Service cluster for analytics.
C. Publish flow logs to Amazon S3 in text format. Use Amazon Athena for analytics.
D. Publish flow logs to Amazon S3 in Apache Parquet format. Use Amazon Athena for analytics.

Correct Answer:

A. Publish flow logs to Amazon CloudWatch Logs. Use Amazon Athena for analytics.

B. Publish flow logs to Amazon CloudWatch Logs. Use an Amazon OpenSearch Service cluster for analytics.

C. Publish flow logs to Amazon S3 in text format. Use Amazon Athena for analytics.

D. Publish flow logs to Amazon S3 in Apache Parquet format. Use Amazon Athena for analytics.

Explanation:

The question is about collecting and analyzing VPC flow logs in a cost-effective manner. Let's break down why option D is the most cost-effective solution:

1. Flow Logs Storage in Amazon S3: - Storing flow logs in Amazon S3 is generally more cost-effective than storing them in Amazon CloudWatch Logs. S3 offers cheaper storage options, especially for large volumes of data which is typical when dealing with flow logs. This is because S3 charges primarily for storage space used, whereas CloudWatch Logs may incur additional costs for data ingestion and retention.

2. Data Format - Apache Parquet: - Apache Parquet is a highly efficient columnar storage file format. When you store data in Parquet format, it reduces the amount of data that needs to be scanned during queries, which can significantly reduce costs and improve performance. This is because Parquet compresses data and allows you to read only the columns that are needed for analysis, rather than entire rows.

3. Analytics with Amazon Athena: - Amazon Athena is a serverless query service that allows you to analyze data stored in Amazon S3 using standard SQL. By querying flow logs stored in Parquet format, Athena can execute queries more efficiently and cost-effectively due to the reduced data scan sizes. Athena charges based on the amount of data scanned by queries, so smaller scan sizes directly translate to lower query costs.

When considering options A and B, both involve storing logs in CloudWatch Logs, which can be more expensive due to data ingestion and storage costs, especially at scale. Option C, while using S3, uses text format, which is less storage-efficient and requires more data to be scanned during queries, leading to higher costs compared to using Parquet format.

Thus, option D is the best choice as it combines the cost-effectiveness of S3 storage with the efficiency of Parquet format and the analytical capabilities of Athena, resulting in a solution that minimizes both storage and analysis costs.

Question # 89

Date: June 15, 2024

A retail company stores transactions, store locations, and customer information tables in four reserved ra3.4xlarge Amazon Redshift cluster nodes. All three tables use even table distribution.

The company updates the store location table only once or twice every few years.

A data engineer notices that Redshift queues are slowing down because the whole store location table is constantly being broadcast to all four compute nodes for most queries. The data engineer wants to speed up the query performance by minimizing the broadcasting of the store location table.

Which solution will meet these requirements in the MOST cost-effective way?

A. Change the distribution style of the store location table from EVEN distribution to ALL distribution.
B. Change the distribution style of the store location table to KEY distribution based on the column that has the highest dimension.
C. Add a join column named store_id into the sort key for all the tables.
D. Upgrade the Redshift reserved node to a larger instance size in the same instance family.

Correct Answer:

A. Change the distribution style of the store location table from EVEN distribution to ALL distribution.

B. Change the distribution style of the store location table to KEY distribution based on the column that has the highest dimension.

C. Add a join column named store_id into the sort key for all the tables.

D. Upgrade the Redshift reserved node to a larger instance size in the same instance family.

Explanation:

The correct answer is A: Change the distribution style of the store location table from EVEN distribution to ALL distribution.

To understand why this is the most cost-effective solution, let's break down the scenario and the options:

1. Current Situation: The retail company uses Amazon Redshift with four ra3.4xlarge nodes, and the store location table is frequently broadcast to all nodes during query execution. This happens because the table uses an EVEN distribution style, which distributes data evenly across all nodes without considering where the other data related to a query resides. This can lead to inefficient queries if data from this table needs to be joined with tables that are distributed differently.

2. Issue: The frequent broadcasting of the store location table to all nodes is causing query performance to degrade. Broadcasting involves sending a full copy of the table to each node, which can be resource-intensive and time-consuming, particularly if the table is large.

3. Option A - ALL Distribution: Changing the distribution style to ALL means that a full copy of the table is stored on each node. This eliminates the need to broadcast the table during query execution because each node already has a copy. The ALL distribution style is especially effective for small, infrequently updated tables that are often joined with other tables, which matches the store location table in this case. Since the table is updated rarely (once or twice every few years), the overhead of maintaining multiple copies is minimal, making this option cost-effective.

4. Option B - KEY Distribution: Changing to a KEY distribution requires choosing a column that can be used to distribute data across nodes based on its values. This can be effective for large tables that frequently join on the distribution key. However, it may not be suitable here because it requires careful selection of the key column and doesn't necessarily solve the issue of constant broadcasting if the joins don't align well with the distribution.

5. Option C - Sort Key: Adding a join column to the sort key optimizes how data is sorted and accessed but does not address the issue of data distribution. It can improve query performance for certain types of queries but won't prevent broadcasting.

6. Option D - Upgrade Node Size: Upgrading to a larger instance size would increase computational power and storage but is not a targeted solution to the specific problem of broadcasting. It would also incur higher costs without necessarily solving the distribution issue.

In summary, option A is the most cost-effective solution because it directly addresses the problem of frequent broadcasting by ensuring that each node has a complete copy of the store location table, thus eliminating the need to broadcast it during queries. This approach leverages the small size and infrequent update pattern of the table, making it an ideal candidate for the ALL distribution style.

Question # 90

Date: June 15, 2024

A company has a data warehouse that contains a table that is named Sales. The company stores the table in Amazon Redshift. The table includes a column that is named city_name. The company wants to query the table to find all rows that have a city_name that starts with "San" or "El".

Which SQL query will meet this requirement?

A. Select * from Sales where city_name ~ ‘$(San|El)*’;
B. Select * from Sales where city_name ~ ‘^(San|El)*’;
C. Select * from Sales where city_name ~’$(San&El;)*’;
D. Select * from Sales where city_name ~ ‘^(San&El;)*’;

Correct Answer:

A. Select * from Sales where city_name ~ ‘$(San|El)*’;

B. Select * from Sales where city_name ~ ‘^(San|El)*’;

C. Select * from Sales where city_name ~’$(San&El;)*’;

D. Select * from Sales where city_name ~ ‘^(San&El;)*’;

Explanation:

The task is to write an SQL query to find all rows in a table named Sales in Amazon Redshift where the column city_name starts with either "San" or "El".

The correct answer is B: `Select from Sales where city_name ~ '^(San|El)';`

Let's break down why this is the correct choice:

1. Understanding the SQL Pattern Matching Operator: - The `~` operator in Redshift is used for pattern matching using POSIX regular expressions. It functions similar to the LIKE operator but is more powerful as it supports complex pattern matching.

2. Regular Expression Details: - In the regular expression `'^(San|El)'`: - The `^` symbol is a special character in regular expressions that anchors the match to the start of the string. This means we're looking for strings that start with the following pattern. - `(San|El)` is a group that uses the pipe `|` symbol to denote an OR condition between "San" and "El". This means the string should start with either "San" or "El". - The `` following the group `(San|El)` is a quantifier that means "zero or more occurrences." However, in this context, the `` is not necessary for the task of simply matching the start because we're only concerned with the initial match, but it doesn't affect the correctness of finding strings that start with "San" or "El".

3. Why Other Options Are Incorrect: - Option A: `Select from Sales where city_name ~ '$(San|El)';` uses `$(San|El)`. The `$` symbol is used to anchor a match to the end of a string in regular expressions, which is opposite to what is needed here. We want to match the start of the string. - Option C: `Select from Sales where city_name ~ '$(San&El;)';` uses a `&` symbol instead of `|`, which is incorrect as `&` is not a valid operator for logical OR in regular expressions. - Option D: `Select from Sales where city_name ~ '^(San&El;)';` similarly uses the incorrect `&` symbol.

Therefore, the correct expression that matches city names starting with either "San" or "El" is found in option B with the use of `^(San|El)`, ensuring the pattern matches only at the beginning of the city_name string.

Question # 91 Date: June 15, 2024

A company needs to send customer call data from its on-premises PostgreSQL database to AWS to generate near real-time insights. The solution must capture and load updates from operational data stores that run in the PostgreSQL database. The data changes continuously.

A data engineer configures an AWS Database Migration Service (AWS DMS) ongoing replication task. The task reads changes in near real time from the PostgreSQL source database transaction logs for each table. The task then sends the data to an Amazon Redshift cluster for processing.

The data engineer discovers latency issues during the change data capture (CDC) of the task. The data engineer thinks that the PostgreSQL source database is causing the high latency.

Which solution will confirm that the PostgreSQL database is the source of the high latency?

A. Use Amazon CloudWatch to monitor the DMS task. Examine the CDCIncomingChanges metric to identify delays in the CDC from the source database.
B. Verify that logical replication of the source database is configured in the postgresql.conf configuration file.

C. Enable Amazon CloudWatch Logs for the DMS endpoint of the source database. Check for error messages.
D. Use Amazon CloudWatch to monitor the DMS task. Examine the CDCLatencySource metric to identify delays in the CDC from the source database.

Correct Answer:

A. Use Amazon CloudWatch to monitor the DMS task. Examine the CDCIncomingChanges metric to identify delays in the CDC from the source database.

B. Verify that logical replication of the source database is configured in the postgresql.conf configuration file.

C. Enable Amazon CloudWatch Logs for the DMS endpoint of the source database. Check for error messages.

D. Use Amazon CloudWatch to monitor the DMS task. Examine the CDCLatencySource metric to identify delays in the CDC from the source database.

Explanation:

The correct answer is D, which involves using Amazon CloudWatch to monitor the AWS Database Migration Service (AWS DMS) task and examining the CDCLatencySource metric to identify delays in the Change Data Capture (CDC) from the source database. Here's why this is the appropriate solution:

1. Understanding the Problem: The scenario describes a situation where there is high latency during the CDC process from a PostgreSQL source database to an Amazon Redshift cluster. The task involves ongoing replication, which means it continuously captures changes to keep the data in sync in near real-time.

2. Identifying Latency: Latency can occur at various stages in a data pipeline. In this context, it's crucial to determine whether the latency is happening at the source (PostgreSQL) or somewhere else in the pipeline. The CDCLatencySource metric specifically measures the time delay between when a change occurs in the source database and when it is captured by AWS DMS. This metric is essential for diagnosing whether the source database is indeed the bottleneck.

3. Using the Correct Metric: - The CDCLatencySource metric provides insights specifically into the replication delay from the source database. It helps you pinpoint whether the PostgreSQL database is causing the issue, as it reflects the time it takes for changes to be read from the source database's transaction logs. - Other metrics, like CDCIncomingChanges, may show the volume of changes but not necessarily help in understanding where the delay is occurring.

4. CloudWatch Monitoring: Amazon CloudWatch is a powerful monitoring tool that allows you to track various metrics related to AWS services. By using CloudWatch to monitor DMS tasks, you gain visibility into the performance and latency of data replication processes.

5. Why Other Options are Less Suitable: - Option A: CDCIncomingChanges shows the number of changes being captured but does not specifically diagnose latency from the source. - Option B: Checking the configuration file for logical replication is more about ensuring that replication is set up correctly, not diagnosing latency. - Option C: Enabling CloudWatch Logs might help identify errors but won't directly pinpoint latency issues related to the source database.

In conclusion, using the CDCLatencySource metric via Amazon CloudWatch provides direct evidence of whether the PostgreSQL source database is the cause of the latency, making it the most effective approach for this specific issue.

Question # 92

Date: June 15, 2024

A lab uses IoT sensors to monitor humidity, temperature, and pressure for a project. The sensors send 100 KB of data every 10 seconds. A downstream process will read the data from an Amazon S3 bucket every 30 seconds.

Which solution will deliver the data to the S3 bucket with the LEAST latency?

A. Use Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose to deliver the data to the S3 bucket. Use the default buffer interval for Kinesis Data Firehose.
B. Use Amazon Kinesis Data Streams to deliver the data to the S3 bucket. Configure the stream to use 5 provisioned shards.
C. Use Amazon Kinesis Data Streams and call the Kinesis Client Library to deliver the data to the S3 bucket. Use a 5 second buffer interval from an application.
D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) and Amazon Kinesis Data Firehose to deliver the data to the S3 bucket. Use a 5 second buffer interval for Kinesis Data Firehose.

Correct Answer:

A. Use Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose to deliver the data to the S3 bucket. Use the default buffer interval for Kinesis Data Firehose.

B. Use Amazon Kinesis Data Streams to deliver the data to the S3 bucket. Configure the stream to use 5 provisioned shards.

C. Use Amazon Kinesis Data Streams and call the Kinesis Client Library to deliver the data to the S3 bucket. Use a 5 second buffer interval from an application.

D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) and Amazon Kinesis Data Firehose to deliver the data to the S3 bucket. Use a 5 second buffer interval for Kinesis Data Firehose.

Explanation:

In this scenario, we need to deliver data generated by IoT sensors to an Amazon S3 bucket with minimal latency. The sensors are sending a significant amount of data (100 KB every 10 seconds), and the data needs to be processed or read every 30 seconds.

Let's break down why option C is the most suitable choice:

1. Amazon Kinesis Data Streams: This service is designed for real-time data streaming. It can handle large volumes of incoming data with minimal delay, making it an excellent choice for scenarios where low latency is crucial. Kinesis Data Streams will efficiently ingest the incoming data from the IoT sensors.

2. Kinesis Client Library (KCL): The KCL is a powerful library used to build applications that can process real-time data from Kinesis Data Streams. By using KCL, you can create a custom application that reads the data from the stream and immediately processes it or forwards it to another destination, such as an S3 bucket.

3. Buffer Interval of 5 Seconds: The use of a 5-second buffer interval in the application ensures that data is processed and transferred to S3 very quickly after it becomes available. This short buffer interval minimizes the delay between data arrival and storage, thus reducing latency significantly.

Option C leverages a combination of Kinesis Data Streams for real-time data ingestion and KCL for immediate processing and delivery, with a very short buffer interval. This setup is optimized for low latency, as it allows the data to be quickly moved from the stream to the S3 bucket within seconds of its arrival.

In contrast, other options (A, B, D) either involve longer buffer intervals or additional processing steps that could introduce delays, making them less efficient for scenarios demanding the least latency. For instance, using Kinesis Data Firehose (as in options A and D) typically involves a buffer interval that can introduce some latency, and option B does not specify an immediate processing mechanism like KCL, which is necessary for low-latency requirements.

Question # 93

Date: June 15, 2024

A company wants to use machine learning (ML) to perform analytics on data that is in an Amazon S3 data lake. The company has two data transformation requirements that will give consumers within the company the ability to create reports.

The company must perform daily transformations on 300 GB of data that is in a variety format that must arrive in Amazon S3 at a scheduled time. The company must perform one-time transformations of terabytes of archived data that is in the S3 data lake. The company uses Amazon Managed Workflows for Apache Airflow (Amazon MWAA) Directed Acyclic Graphs (DAGs) to orchestrate processing.

Which combination of tasks should the company schedule in the Amazon MWAA DAGs to meet these requirements MOST cost-effectively? (Choose two.)

A. For daily incoming data, use AWS Glue crawlers to scan and identify the schema.

B. For daily incoming data, use Amazon Athena to scan and identify the schema.
C. For daily incoming data, use Amazon Redshift to perform transformations.
D. For daily and archived data, use Amazon EMR to perform data transformations.
E. For archived data, use Amazon SageMaker to perform data transformations.

Correct Answer:

A. For daily incoming data, use AWS Glue crawlers to scan and identify the schema.

B. For daily incoming data, use Amazon Athena to scan and identify the schema.

C. For daily incoming data, use Amazon Redshift to perform transformations.

D. For daily and archived data, use Amazon EMR to perform data transformations.

E. For archived data, use Amazon SageMaker to perform data transformations.

Explanation:

The correct answer to the question is "A" and "D", and here's why:

A. For daily incoming data, use AWS Glue crawlers to scan and identify the schema.

AWS Glue is a fully managed ETL (Extract, Transform, Load) service that makes it easy to prepare and load data for analytics. Glue Crawlers are specifically designed to automatically scan data stored in Amazon S3 and determine its schema, which is crucial when dealing with data in a variety of formats. This automation simplifies the process of schema discovery and ensures that the data is properly cataloged for further analysis and transformation. Since the data is arriving in Amazon S3 at a scheduled time, using AWS Glue crawlers is a cost-effective and efficient way to handle schema identification on a daily basis, without the need for manual intervention.

D. For daily and archived data, use Amazon EMR to perform data transformations.

Amazon EMR (Elastic MapReduce) is a cloud-native big data platform that provides a managed Hadoop framework to process and analyze vast amounts of data using the Apache Hadoop ecosystem and tools like Apache Spark. It is highly scalable, allowing users to process data at any scale, which is ideal for both daily transformations of 300 GB and one-time transformations of terabytes of archived data. EMR is cost-effective because it allows you to pay only for the resources you use, and you can optimize costs further by utilizing spot instances. Additionally, EMR integrates well with other AWS services and provides flexibility to run custom scripts for comprehensive data transformations. Using EMR for both daily and archived data ensures a consistent and scalable transformation process.

Why Other Options Are Less Suitable:

- B. For daily incoming data, use Amazon Athena to scan and identify the schema. While Amazon Athena is excellent for querying data directly in S3 using SQL without the need for complex ETL processes, it is not designed to automatically identify or manage schema changes. Athena is more suitable for ad-hoc querying rather than scheduled schema identification tasks.

- C. For daily incoming data, use Amazon Redshift to perform transformations. Amazon Redshift is a data warehouse, optimized for complex queries across large datasets, but it's not the most cost-effective choice for ETL processes on data arriving daily in S3. Importing data into Redshift for transformation adds unnecessary complexity and cost, especially when dealing with variable schema data.

- E. For archived data, use Amazon SageMaker to perform data transformations. Amazon SageMaker is primarily a machine learning service designed to build, train, and deploy ML models. It is not optimized for performing large-scale data transformations, particularly on archive data. Using SageMaker for transformations would be inefficient and costly, as it's not tailored for this specific task.

Thus, combining AWS Glue for schema detection and Amazon EMR for data transformations offers a robust, cost-effective solution to meet the company's requirements.

Question # 94

Date: August 12, 2024

A data engineer is using an AWS Glue crawler to catalog data that is in an Amazon S3 bucket. The S3 bucket contains both .csv and json files. The data engineer configured the crawler to exclude the .json files from the catalog.

When the data engineer runs queries in Amazon Athena, the queries also process the excluded .json files. The data engineer wants to resolve this issue. The data engineer needs a solution that will not affect access requirements for the .csv files in the source S3 bucket.

Which solution will meet this requirement with the SHORTEST query times?

A. Adjust the AWS Glue crawler settings to ensure that the AWS Glue crawler also excludes .json files.
B. Use the Athena console to ensure the Athena queries also exclude the .json files.
C. Relocate the .json files to a different path within the S3 bucket.
D. Use S3 bucket policies to block access to the .json files.

Correct Answer:

A. Adjust the AWS Glue crawler settings to ensure that the AWS Glue crawler also excludes .json files.

B. Use the Athena console to ensure the Athena queries also exclude the .json files.

C. Relocate the .json files to a different path within the S3 bucket.

D. Use S3 bucket policies to block access to the .json files.

Explanation:

The correct answer is C: Relocate the .json files to a different path within the S3 bucket.

Here's why this solution is effective:

1. AWS Glue Crawler Configuration: AWS Glue crawlers are used to catalog data in S3 by creating metadata tables in the AWS Glue Data Catalog. When a crawler is configured to exclude certain file types, it typically means these files won't be included in the table schema. However, if the files are still in the same location, they can still be accessed directly through other means, such as through queries in Amazon Athena, if such queries are not strictly defined to exclude them.

2. Issue with Current Setup: The problem here is that even though the AWS Glue crawler is configured to exclude the `.json` files, the Athena queries are still processing them. This suggests that the crawler's exclusion only affects the Glue Data Catalog and not the Athena query operations if the queries are not correctly scoped to exclude these files. Athena reads directly from S3, and without any filtering logic in the query, it will process all files in the queried location.

3. Why Relocation Works Best: Moving the `.json` files to a separate directory or path in the S3 bucket ensures that Athena queries targeting the original path cannot inadvertently include these files, as they are physically separated from the `.csv` files. This physical separation ensures that even if a query targets all files in the original directory, it will not include the `.json` files, thus reducing query processing time and cost because it eliminates unnecessary data processing.

4. Query Performance: By relocating the `.json` files, you effectively reduce the amount of data that Athena needs to read and process. Athena is charged based on the amount of data scanned by queries, so reducing this by excluding unnecessary files (like `.json` in this case) will lead to shorter query times and lower costs.

5. No Impact on Access Requirements: This approach does not affect access permissions or requirements for the `.csv` files. They remain in their original directory, ensuring that any processes or users accessing these files continue to do so without any additional configuration changes.

Option A would not solve the problem of Athena queries accessing `.json` files because it only affects the Glue catalog, not Athena queries directly. Option B would require ensuring that every Athena query is correctly configured to exclude `.json` files, which can be error-prone and not efficient. Option D could lead to access issues if `.json` files are needed for other purposes.

Overall, relocating the `.json` files is a straightforward solution that directly addresses the problem by physically separating the files, ensuring that Athena queries are efficient and only process the intended data.

Question # 95

Date: June 15, 2024

An insurance company stores transaction data that the company compressed with gzip.

The company needs to query the transaction data for occasional audits.

Which solution will meet this requirement in the MOST cost-effective way?

A. Store the data in Amazon Glacier Flexible Retrieval. Use Amazon S3 Glacier Select to query the data.
B. Store the data in Amazon S3. Use Amazon S3 Select to query the data.
C. Store the data in Amazon S3. Use Amazon Athena to query the data.
D. Store the data in Amazon Glacier Instant Retrieval. Use Amazon Athena to query the data.

Correct Answer:

A. Store the data in Amazon Glacier Flexible Retrieval. Use Amazon S3 Glacier Select to query the data.

B. Store the data in Amazon S3. Use Amazon S3 Select to query the data.

C. Store the data in Amazon S3. Use Amazon Athena to query the data.

D. Store the data in Amazon Glacier Instant Retrieval. Use Amazon Athena to query the data.

Explanation:

The correct answer is B: Store the data in Amazon S3. Use Amazon S3 Select to query the data.

Let's break down why this solution is the most cost-effective and efficient for the insurance company's needs:

1. Amazon S3 Storage: - Amazon S3 (Simple Storage Service) is a highly scalable, durable, and cost-effective object storage service. It is ideal for storing large amounts of data, such as transaction logs, and offers easy accessibility and retrieval. - S3 allows for storing data in various storage classes, which can be tailored to the access needs (e.g., Standard, Intelligent-Tiering, Infrequent Access).

2. Amazon S3 Select: - S3 Select is a feature that allows you to retrieve a subset of data from an object by using simple SQL expressions. This means you can perform queries directly on the data stored in S3 without having to download the entire object. - S3 Select is particularly effective for querying compressed files like those compressed with gzip, as it can read and process the data in its compressed form, thus reducing the amount of data transferred and processed, saving both time and costs. - This makes it ideal for occasional audits, where only specific portions of the data need to be queried, rather than processing the entire dataset.

3. Cost-Effectiveness: - Compared to Amazon Glacier, S3 provides more immediate access to data, which is crucial for audit purposes where data might need to be accessed promptly. - Glacier is designed for long-term archival and is typically cheaper for storage costs but incurs additional costs and time delays for data retrieval. - S3 Select allows for efficient querying without the need for additional services like Athena, keeping costs lower since it minimizes data scanned and processed.

In summary, storing data in Amazon S3 and using S3 Select provides a balance of cost, accessibility, and performance, specifically suited for occasional queries and audits. It avoids the latency and potential costs associated with Glacier retrievals and additional processing services, making it the most cost-effective option for the given scenario.

Question # 96

Date: June 14, 2024

A data engineer finished testing an Amazon Redshift stored procedure that processes and inserts data into a table that is not mission critical. The engineer wants to automatically run the stored procedure on a daily basis.

Which solution will meet this requirement in the MOST cost-effective way?

A. Create an AWS Lambda function to schedule a cron job to run the stored procedure.
B. Schedule and run the stored procedure by using the Amazon Redshift Data API in an Amazon EC2 Spot Instance.
C. Use query editor v2 to run the stored procedure on a schedule.
D. Schedule an AWS Glue Python shell job to run the stored procedure.

Correct Answer:

A. Create an AWS Lambda function to schedule a cron job to run the stored procedure.

B. Schedule and run the stored procedure by using the Amazon Redshift Data API in an Amazon EC2 Spot Instance.

C. Use query editor v2 to run the stored procedure on a schedule.

D. Schedule an AWS Glue Python shell job to run the stored procedure.

Explanation:

The correct answer is C: Use query editor v2 to run the stored procedure on a schedule.

Here's why this is the most appropriate and cost-effective solution:

1. Purpose of Query Editor v2: Amazon Redshift's query editor v2 is an integrated tool that allows users to connect to their Redshift clusters, run SQL queries, and manage their data without needing a separate client tool. With the introduction of query editor v2, AWS has added features to schedule query executions directly from the interface.

2. No Additional Infrastructure: By using query editor v2, you're utilizing built-in functionality that doesn't require setting up and managing additional AWS services or infrastructure. This is inherently more cost-effective because there are no additional AWS resources like EC2 instances or Lambda functions to manage or incur costs from.

3. Ease of Use and Management: Query editor v2 provides a simple and user-friendly interface to schedule queries, including stored procedures, to run at specified intervals. This reduces the complexity involved in setting up a recurring task using other AWS services.

4. Cost-Effective: Since you're leveraging a feature of Amazon Redshift itself without needing to deploy, manage, or pay for additional compute resources, this approach is more cost-effective. You avoid the costs associated with running EC2 instances or the execution costs of AWS Lambda functions.

5. Simplicity and Direct Integration: The integration is direct and straightforward, reducing the potential for errors that could arise from more complex solutions involving multiple AWS services.

In contrast, the other options involve using additional AWS services (Lambda, EC2, Glue) that introduce extra management overhead and potential costs, which makes them less cost-effective for this use case. In particular:

- Option A: Using AWS Lambda would involve setting up a Lambda function and possibly CloudWatch Events for scheduling, leading to additional costs and management overhead. - Option B: Running tasks on an EC2 Spot Instance might save costs compared to on-demand instances but still involves managing an EC2 instance and the associated complexities. - Option D: AWS Glue is primarily designed for ETL tasks, and while it could be used to run a stored procedure, it is overkill for this simple requirement and incurs costs related to Glue jobs.

Thus, using query editor v2 to schedule the stored procedure is the simplest, most integrated, and cost-effective solution.

Question # 97

Date: June 15, 2024

A marketing company collects clickstream data. The company sends the clickstream data to Amazon Kinesis Data Firehose and stores the clickstream data in Amazon S3. The company wants to build a series of dashboards that hundreds of users from multiple departments will use.

The company will use Amazon QuickSight to develop the dashboards. The company wants a solution that can scale and provide daily updates about clickstream activity.

Which combination of steps will meet these requirements MOST cost-effectively? (Choose two.)

A. Use Amazon Redshift to store and query the clickstream data.
B. Use Amazon Athena to query the clickstream data
C. Use Amazon S3 analytics to query the clickstream data.
D. Access the query data through a QuickSight direct SQL query.
E. Access the query data through QuickSight SPICE (Super-fast, Parallel, In-memory Calculation Engine). Configure a daily refresh for the dataset.

Correct Answer:

A. Use Amazon Redshift to store and query the clickstream data.

B. Use Amazon Athena to query the clickstream data

C. Use Amazon S3 analytics to query the clickstream data.

D. Access the query data through a QuickSight direct SQL query.

E. Access the query data through QuickSight SPICE (Super-fast, Parallel, In-memory Calculation Engine). Configure a daily refresh for the dataset.

Explanation:

The scenario involves a marketing company that collects clickstream data and stores it in Amazon S3 using Amazon Kinesis Data Firehose. The company needs to build dashboards using Amazon QuickSight that are scalable and provide daily updates for hundreds of users from multiple departments. The goal is to achieve this in a cost-effective manner. Let's break down why options B and E are the most suitable choices:

B. Use Amazon Athena to query the clickstream data:

Amazon Athena is an interactive query service that allows you to analyze data directly in Amazon S3 using standard SQL. It is a serverless solution, meaning there is no infrastructure to manage, and you only pay for the queries you run, making it very cost-effective for ad-hoc querying. Since the clickstream data is already stored in S3, Athena can directly query this data without needing to move it to another service. This eliminates the need for additional storage or data transfer costs. Athena integrates seamlessly with QuickSight, allowing you to create datasets for your dashboards. Its pay-per-query pricing model aligns well with the cost-effective requirement.

E. Access the query data through QuickSight SPICE (Super-fast, Parallel, In-memory Calculation Engine). Configure a daily refresh for the dataset:

QuickSight SPICE is an in-memory calculation engine that allows you to perform fast analyses and visualizations. By using SPICE, you can handle large datasets and perform quick queries, which is crucial when hundreds of users are accessing the dashboards. SPICE can ingest data from Athena, and by configuring a daily refresh, it ensures that the dashboards are updated with the latest clickstream data without manual intervention. This setup optimizes performance and scalability, as SPICE can efficiently handle concurrent user access. Additionally, SPICE is included in the QuickSight subscription and offers a certain amount of capacity at no additional charge, contributing to the cost-effectiveness of the solution.

In contrast, using Amazon Redshift (option A) would involve setting up a data warehouse, which could be more costly and complex compared to Athena's serverless model. Amazon S3 analytics (option C) is not designed for direct querying or integration with QuickSight, and using direct SQL queries (option D) in QuickSight might not provide the same performance benefits as SPICE, especially when scaling to hundreds of users. Hence, the combination of Athena and SPICE in QuickSight effectively meets the company's requirements for scalability, cost-efficiency, and timely data updates.

Question # 98

Date: June 15, 2024

A data engineer is building a data orchestration workflow. The data engineer plans to use a hybrid model that includes some on-premises resources and some resources that are in the cloud. The data engineer wants to prioritize portability and open source resources.

Which service should the data engineer use in both the on-premises environment and the cloud-based environment?

A. AWS Data Exchange
B. Amazon Simple Workflow Service (Amazon SWF)
C. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
D. AWS Glue

Correct Answer:

A. AWS Data Exchange

B. Amazon Simple Workflow Service (Amazon SWF)

C. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

D. AWS Glue

Explanation:

The correct answer is C: Amazon Managed Workflows for Apache Airflow (Amazon MWAA).

Here's why this is the appropriate choice for the scenario described:

1. Portability and Open Source: The data engineer emphasizes the importance of portability and open source resources. Apache Airflow is an open-source tool for orchestrating complex computational workflows and data processing pipelines. It is widely used in the industry due to its flexibility and ability to integrate with various systems. By selecting an open-source tool like Apache Airflow, the engineer ensures that the workflow is not tied to a specific vendor's ecosystem and can be easily migrated or adapted as needed.

2. Hybrid Model Support: The engineer needs a solution that works seamlessly in a hybrid environment, incorporating both on-premises and cloud-based resources. Apache Airflow can be deployed on-premises, and Amazon Managed Workflows for Apache Airflow (Amazon MWAA) is the AWS-managed service version of Airflow that runs in the cloud. This dual capability allows workflows to be managed consistently across different environments, providing flexibility and ease of integration for hybrid setups.

3. AWS Integration: By using Amazon MWAA, the engineer benefits from tight integration with AWS services. This makes it easier to manage workflows that involve AWS resources while still maintaining the ability to run components on-premises using standard Apache Airflow installations.

4. Managed Service: Amazon MWAA manages the underlying infrastructure for running Apache Airflow, simplifying the operational overhead. This allows the data engineer to focus more on building and optimizing workflows rather than managing infrastructure.

In contrast, the other options do not meet the requirements as effectively: - AWS Data Exchange (A) is primarily a marketplace for data providers and consumers and doesn't facilitate workflow orchestration. - Amazon Simple Workflow Service (Amazon SWF) (B) is a cloud-native workflow service but is not open source and does not offer the portability that Apache Airflow does. - AWS Glue (D) is a fully managed ETL service, which is more focused on data transformation and not specifically designed for orchestrating diverse workflows across hybrid environments.

Thus, Amazon MWAA is the optimal choice for the data engineer's needs in terms of portability, open source compatibility, and hybrid model support.

Question # 99

Date: June 14, 2024

A gaming company uses a NoSQL database to store customer information. The company is planning to migrate to AWS.

The company needs a fully managed AWS solution that will handle high online transaction processing (OLTP) workload, provide single-digit millisecond performance, and provide high availability around the world.

Which solution will meet these requirements with the LEAST operational overhead?

A. Amazon Keyspaces (for Apache Cassandra)
B. Amazon DocumentDB (with MongoDB compatibility)
C. Amazon DynamoDB
D. Amazon Timestream

Correct Answer:

A. Amazon Keyspaces (for Apache Cassandra)

B. Amazon DocumentDB (with MongoDB compatibility)

C. Amazon DynamoDB

D. Amazon Timestream

Explanation:
The correct answer is C: Amazon DynamoDB.

Here's why Amazon DynamoDB is the best fit for the requirements described in the question:

1. Fully Managed NoSQL Database: Amazon DynamoDB is a fully managed NoSQL database service provided by AWS. This means that AWS takes care of the operational aspects such as maintenance, patching, and scaling, allowing the company to focus on application development rather than infrastructure management.

2. High Performance: DynamoDB is specifically designed to handle high-volume OLTP workloads with low latency. It is capable of delivering single-digit millisecond response times for reads and writes, which is crucial for applications like gaming where real-time interactions are key.

3. Global Availability: DynamoDB offers a feature called Global Tables, which allows tables to be replicated across multiple AWS Regions. This ensures high availability and low-latency access to data for users around the world, which is perfect for a gaming company with a global user base.

4. Scalability: DynamoDB can automatically scale up and down to adjust for the capacity requirements of the application. This is particularly beneficial for a gaming company that might experience fluctuating traffic patterns, ensuring that performance remains consistent regardless of the workload.

5. Minimal Operational Overhead: Since DynamoDB is fully managed and offers features like automatic scaling and global replication, it requires minimal operational effort to maintain. This reduces the need for a large operational staff to manage database infrastructure, aligning with the company's requirement for minimal operational overhead.

In contrast, let's briefly look at why the other options are not as suitable:

- Amazon Keyspaces: While it is a managed service for Apache Cassandra and can handle large workloads, it is not as deeply integrated into AWS as DynamoDB and might require more operational work in terms of configuration and management.

- Amazon DocumentDB: This service is geared towards applications requiring compatibility with MongoDB, which might not be necessary or optimal for the company's needs. Additionally, DocumentDB is more focused on document-based data models rather than the key-value or wide-column store models.

- Amazon Timestream: This is a managed time-series database service designed for IoT and operational applications that require time-series data management. It is not suitable for the OLTP workloads typically associated with gaming applications.

Overall, Amazon DynamoDB is tailored to meet the high performance, global availability, and low operational overhead requirements specified by the gaming company in the question.

Question # 100

Date: June 15, 2024

A data engineer creates an AWS Lambda function that an Amazon EventBridge event will invoke. When the data engineer tries to invoke the Lambda function by using an EventBridge event, an AccessDeniedException message appears.

How should the data engineer resolve the exception?

A. Ensure that the trust policy of the Lambda function execution role allows EventBridge to assume the execution role.
B. Ensure that both the IAM role that EventBridge uses and the Lambda function's resource-based policy have the necessary permissions.
C. Ensure that the subnet where the Lambda function is deployed is configured to be a private subnet.
D. Ensure that EventBridge schemas are valid and that the event mapping configuration is correct.

Correct Answer:

A. Ensure that the trust policy of the Lambda function execution role allows EventBridge to assume the execution role.

B. Ensure that both the IAM role that EventBridge uses and the Lambda function's resource-based policy have the necessary permissions.

C. Ensure that the subnet where the Lambda function is deployed is configured to be a private subnet.

D. Ensure that EventBridge schemas are valid and that the event mapping configuration is correct.

Explanation:

The correct answer is B: Ensure that both the IAM role that EventBridge uses and the Lambda function's resource-based policy have the necessary permissions.

Here's why this is the right approach:

1. Understanding the Error: - The `AccessDeniedException` suggests there is a permissions issue. When integrating AWS services, permissions must be configured correctly for both the service invoking the action (EventBridge) and the service being invoked (Lambda).

2. AWS Lambda and EventBridge Interaction: - AWS EventBridge (formerly CloudWatch Events) is a service that allows you to respond to state changes in your AWS resources. It can invoke AWS Lambda functions, but to do so, the function must explicitly allow EventBridge to execute it.

3. Permissions Required: - IAM Role for EventBridge: EventBridge needs permissions to invoke the Lambda function. These permissions are typically set in an IAM role that EventBridge assumes. - Lambda Resource-Based Policy: Lambda functions also have resource-based policies that specify which AWS services or principals can invoke them. This policy must include permissions for EventBridge to invoke the function.

4. Solving the AccessDeniedException: - Step 1: Check the IAM role that EventBridge uses to ensure it has permissions to invoke the Lambda function. This role should have an appropriate policy that allows the `lambda:InvokeFunction` action. - Step 2: Update the Lambda function's resource-based policy to explicitly allow invocation by EventBridge. This involves adding a statement that grants the `events.amazonaws.com` service permission to invoke the function.

5. Why Other Options Are Incorrect: - Option A: Only mentions the trust policy of the Lambda execution role. Trust policies define which principals can assume the role, which is not directly related to the issue of EventBridge invoking Lambda. - Option C: Involves subnet configuration, which is unrelated to permission errors. Subnet configurations deal with network access, not with IAM policies or permissions. - Option D: Refers to schema validation and event mapping, which are not related to permission issues. An `AccessDeniedException` is strictly a permissions problem, not a data format or configuration issue.

By ensuring that both the EventBridge IAM role and the Lambda function's resource-based policy have the necessary permissions, the data engineer addresses the root cause of the AccessDeniedException, allowing EventBridge to successfully invoke the Lambda function.

Question # 101

Date: June 15, 2024

A company uses a data lake that is based on an Amazon S3 bucket. To comply with regulations, the company must apply two layers of server-side encryption to files that are uploaded to the S3 bucket. The company wants to use an AWS Lambda function to apply the necessary encryption.

Which solution will meet these requirements?

A. Use both server-side encryption with AWS KMS keys (SSE-KMS) and the Amazon S3 Encryption Client.
B. Use dual-layer server-side encryption with AWS KMS keys (DSSE-KMS).
C. Use server-side encryption with customer-provided keys (SSE-C) before files are uploaded.
D. Use server-side encryption with AWS KMS keys (SSE-KMS).

Correct Answer:

A. Use both server-side encryption with AWS KMS keys (SSE-KMS) and the Amazon S3 Encryption Client.

B. Use dual-layer server-side encryption with AWS KMS keys (DSSE-KMS).

C. Use server-side encryption with customer-provided keys (SSE-C) before files are uploaded.

D. Use server-side encryption with AWS KMS keys (SSE-KMS).

Explanation:

The requirement here is to apply two layers of server-side encryption to files uploaded to an Amazon S3 bucket. The correct answer is B, "Use dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)."

Let's break down why this is the correct choice:

1. Understanding Dual-Layer Encryption: When regulations require two layers of encryption, it means that the data needs to be encrypted twice, using two distinct encryption processes. This is different from having just a single layer of encryption where data is encrypted only once.

2. AWS KMS and Encryption: AWS KMS (Key Management Service) is a managed service that makes it easy to create and control the encryption keys used to encrypt your data. It integrates with several AWS services, including Amazon S3, allowing you to encrypt data stored in S3 using keys managed by KMS.

3. Dual-Layer Server-Side Encryption with KMS (DSSE-KMS): This feature allows you to apply two layers of encryption using AWS KMS keys. The first layer of encryption is applied before the data is stored in S3, and the second layer is applied once the data is stored. This ensures that even if one layer is compromised, the second layer provides additional security. DSSE-KMS is specifically designed to meet the needs of organizations that must comply with strict regulatory requirements demanding dual encryption.

4. Why Other Options Are Not Suitable: - Option A (SSE-KMS and Encryption Client): This option involves using the Amazon S3 Encryption Client, which is a client-side encryption library. It doesn't inherently provide two layers of server-side encryption as required by the question. - Option C (SSE-C before upload): SSE-C involves customer-provided keys, but it doesn’t specify a two-layer server-side encryption process. It also requires the customer to manage their own encryption keys, which does not automatically provide the dual-layer encryption as required. - Option D (SSE-KMS): This provides a single layer of encryption using KMS keys but does not fulfill the requirement for two layers of encryption.

Therefore, option B is the correct choice as it directly addresses the requirement for dual-layer encryption using AWS KMS, aligning with regulatory needs for enhanced security.

Question # 102

Date: June 15, 2024

A data engineer notices that Amazon Athena queries are held in a queue before the queries run.

How can the data engineer prevent the queries from queueing?

A. Increase the query result limit.
B. Configure provisioned capacity for an existing workgroup.
C. Use federated queries.
D. Allow users who run the Athena queries to an existing workgroup.

Correct Answer:

A. Increase the query result limit.

B. Configure provisioned capacity for an existing workgroup.

C. Use federated queries.

D. Allow users who run the Athena queries to an existing workgroup.

Explanation:

Certainly! The correct answer to the question is option B: "Configure provisioned capacity for an existing workgroup."

Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. When running queries in Athena, they are processed in workgroups. A workgroup is essentially a set of configurations and resources where Athena queries are executed, and it allows for better query management and cost control.

When multiple queries are submitted to Athena, they are handled by the underlying resources allocated to your account. If the demand for queries exceeds the available capacity, the queries can be queued, leading to delays. This is where the concept of "provisioned capacity" comes into play.

Provisioned capacity in Athena allows you to reserve dedicated query processing capacity. By configuring provisioned capacity for a workgroup, you ensure that a certain amount of resources is always available for that workgroup's queries. This means that even during peak times when multiple queries are being submitted concurrently, the queries in that workgroup have a guaranteed amount of processing power, reducing the chances of them being queued.

Option B, therefore, is the correct choice because by configuring provisioned capacity for an existing workgroup, the data engineer can ensure that there is enough dedicated processing capacity for queries, minimizing the likelihood of them being held in a queue before execution.

Let's briefly discuss why the other options are not suitable:

A. Increasing the query result limit does not impact the queuing of queries. The query result limit pertains to the size of the results that can be returned by a single query, not the processing capacity or throughput of queries.

C. Using federated queries allows you to query data across different data sources and locations. While this can be powerful for accessing diverse data, it does not directly address the issue of queries being queued due to insufficient processing capacity.

D. Allowing users to an existing workgroup affects who can execute queries but does not change the capacity available for processing those queries. It may even exacerbate queuing issues if more users are running queries without increasing the available capacity.

In summary, configuring provisioned capacity ensures that your queries have the necessary resources to run promptly, effectively preventing them from being queued.

Question # 103

Date: June 29, 2024

A data engineer needs to debug an AWS Glue job that reads from Amazon S3 and writes to Amazon Redshift. The data engineer enabled the bookmark feature for the AWS Glue job.

The data engineer has set the maximum concurrency for the AWS Glue job to 1.

The AWS Glue job is successfully writing the output to Amazon Redshift. However, the Amazon S3 files that were loaded during previous runs of the AWS Glue job are being reprocessed by subsequent runs.

What is the likely reason the AWS Glue job is reprocessing the files?

A. The AWS Glue job does not have the s3:GetObjectAcl permission that is required for bookmarks to work correctly.
B. The maximum concurrency for the AWS Glue job is set to 1.
C. The data engineer incorrectly specified an older version of AWS Glue for the Glue job.
D. The AWS Glue job does not have a required commit statement.

Correct Answer:

A. The AWS Glue job does not have the s3:GetObjectAcl permission that is required for bookmarks to work correctly.

B. The maximum concurrency for the AWS Glue job is set to 1.

C. The data engineer incorrectly specified an older version of AWS Glue for the Glue job.

D. The AWS Glue job does not have a required commit statement.

Explanation:

The scenario described involves a data engineer who is experiencing an unexpected behavior with an AWS Glue job. This Glue job is designed to read data from Amazon S3 and write it to Amazon Redshift, and the engineer has enabled the bookmark feature to prevent reprocessing of the same data in subsequent runs. Despite this, the job is reprocessing files that were already processed in previous runs.

Let's break down why option D, "The AWS Glue job does not have a required commit statement," is likely the correct answer:

1. AWS Glue Bookmark Feature: Bookmarks in AWS Glue are used to keep track of the data that has been processed in previous job runs. This feature helps ensure that data is not reprocessed unnecessarily, which is particularly useful in incremental ETL operations.

2. Commit Statement: In the context of AWS Glue jobs, a commit statement is typically necessary to finalize the transaction or the processing state of data. Without a proper commit, the job might not update the bookmark correctly, meaning that the system is unaware that the data has already been processed. This would result in the same data being processed again in subsequent runs.

3. Understanding of Reprocessing: The issue at hand is that files are being reprocessed even though bookmarks are enabled. This suggests that the bookmark state is not being updated correctly, which can often be due to a missing or improperly executed commit statement within the job's logic.

4. Other Options Analysis: - Option A: The lack of `s3:GetObjectAcl` permission would affect the ability to read object ACLs but is unlikely to affect bookmark functionality directly. - Option B: Maximum concurrency set to 1 means only one instance of the job runs at a time. This setting limits parallel processing but does not impact the functioning of bookmarks. - Option C: Using an older version of AWS Glue might impact some features but should not inherently cause bookmarks to malfunction unless the version is unsupported or deprecated, which isn't indicated here.

In summary, the most plausible explanation for the job's current behavior is that the AWS Glue job lacks a necessary commit statement to successfully update the bookmark state. This prevents the system from recognizing that the data has already been processed, leading to reprocessing in subsequent runs. Hence, option D is the correct choice.

Question # 104

Date: June 29, 2024

An ecommerce company wants to use AWS to migrate data pipelines from an on-premises environment into the AWS Cloud. The company currently uses a third-party tool in the on-premises environment to orchestrate data ingestion processes.

The company wants a migration solution that does not require the company to manage servers. The solution must be able to orchestrate Python and Bash scripts. The solution must not require the company to refactor any code.

Which solution will meet these requirements with the LEAST operational overhead?

A. AWS Lambda
B. Amazon Managed Workflows for Apache Airflow (Amazon MVVAA)
C. AWS Step Functions
D. AWS Glue

Correct Answer:

A. AWS Lambda

B. Amazon Managed Workflows for Apache Airflow (Amazon MVVAA)

C. AWS Step Functions

D. AWS Glue

Explanation:

The correct answer is B: Amazon Managed Workflows for Apache Airflow (Amazon MWAA).

Let's break down why Amazon MWAA is the best choice given the requirements:

1. Serverless and Managed: The company wants a solution that doesn't require managing servers. Amazon MWAA is a managed service, meaning AWS takes care of the underlying infrastructure. It automatically scales and handles the operational burden, which minimizes the need for the company to manage servers.

2. Orchestration of Python and Bash Scripts: The company's current setup involves orchestrating data ingestion processes using a third-party tool. Amazon MWAA supports Apache Airflow, which is a popular open-source tool for orchestrating complex workflows. Airflow workflows, or DAGs (Directed Acyclic Graphs), can run tasks written in Python, and these tasks can call Bash scripts as needed. This makes it straightforward to run existing Python and Bash scripts without modification.

3. No Code Refactoring Required: The company wants to avoid refactoring any code during the migration. Airflow's flexibility in task management means it can incorporate existing scripts directly into its workflows, allowing the company to continue using their current scripts with minimal changes, if any.

4. Operational Overhead: Amazon MWAA reduces operational overhead by providing a fully managed environment for running Apache Airflow. It automatically handles patching, scaling, and backend infrastructure, which aligns with the requirement of having the least operational overhead.

Now, let's briefly consider why the other options are less suitable:

- AWS Lambda: While AWS Lambda is serverless and can execute Python and Bash scripts, it primarily focuses on running short-lived tasks and doesn't inherently provide orchestration capabilities for complex workflows. It would require additional management and setup to orchestrate multiple steps, increasing operational overhead.

- AWS Step Functions: Step Functions is another serverless orchestration service, but it is generally more suited for integrating AWS services. While it can orchestrate tasks, integrating existing Python and Bash scripts would likely require more effort and potential code changes compared to using Airflow, where such scripts can be more seamlessly incorporated.

- AWS Glue: AWS Glue is a serverless data integration service primarily designed for ETL (Extract, Transform, Load) jobs. While it can run Python scripts, it is not primarily designed for orchestrating complex workflows involving multiple types of tasks, making it less suitable for this scenario.

In summary, Amazon MWAA is the best fit as it meets all the requirements: it's serverless, can orchestrate existing scripts with minimal changes, and incurs the least operational overhead.

Question # 105

Date: June 29, 2024

A retail company stores data from a product lifecycle management (PLM) application in an on-premises MySQL database. The PLM application frequently updates the database when transactions occur.

The company wants to gather insights from the PLM application in near real time. The company wants to integrate the insights with other business datasets and to analyze the combined dataset by using an Amazon Redshift data warehouse.

The company has already established an AWS Direct Connect connection between the on-premises infrastructure and AWS.

Which solution will meet these requirements with the LEAST development effort?

A. Run a scheduled AWS Glue extract, transform, and load (ETL) job to get the MySQL database updates by using a Java Database Connectivity (JDBC) connection. Set Amazon Redshift as the destination for the ETL job.
B. Run a full load plus CDC task in AWS Database Migration Service (AWS DMS) to continuously replicate the MySQL database changes. Set Amazon Redshift as the destination for the task.
C. Use the Amazon AppFlow SDK to build a custom connector for the MySQL database to continuously replicate the database changes. Set Amazon Redshift as the destination for the connector.
D. Run scheduled AWS DataSync tasks to synchronize data from the MySQL database. Set Amazon Redshift as the destination for the tasks.

Correct Answer:

A. Run a scheduled AWS Glue extract, transform, and load (ETL) job to get the MySQL database updates by using a Java Database Connectivity (JDBC) connection. Set Amazon Redshift as the destination for the ETL job.

B. Run a full load plus CDC task in AWS Database Migration Service (AWS DMS) to continuously replicate the MySQL database changes. Set Amazon Redshift as the destination for the task.

C. Use the Amazon AppFlow SDK to build a custom connector for the MySQL database to continuously replicate the database changes. Set Amazon Redshift as the destination for the connector.

D. Run scheduled AWS DataSync tasks to synchronize data from the MySQL database. Set Amazon Redshift as the destination for the tasks.

Explanation:

The correct answer is B: Run a full load plus CDC task in AWS Database Migration Service (AWS DMS) to continuously replicate the MySQL database changes. Set Amazon Redshift as the destination for the task.

Here's why this is the best solution with the least development effort:

1. Continuous Data Capture (CDC): AWS DMS supports continuous data capture, which means it can replicate ongoing changes from the MySQL database in near real time. This capability is crucial for the company's requirement to gather insights in near real time. CDC ensures that as soon as there are updates in the MySQL database, these changes are captured and replicated to the target data warehouse, Amazon Redshift.

2. Full Load and CDC Task: The combination of a full load and CDC task allows AWS DMS to initially load the entire database from MySQL to Redshift and then continuously apply changes. This approach is efficient and minimizes the manual effort required to keep the data synchronized.

3. Minimal Development Effort: AWS DMS is designed to be a low-code solution for database migration and replication. It offers an out-of-the-box service that doesn't require extensive custom development, unlike a custom solution that might involve the Amazon AppFlow SDK or building custom ETL pipelines.

4. Direct Connect Integration: Since the company already has AWS Direct Connect in place, it provides a dedicated network connection between the on-premises infrastructure and AWS, which can be leveraged by AWS DMS for efficient and reliable data transfer.

5. Ease of Use and Flexibility: AWS DMS is a managed service, which means AWS handles the operational maintenance, scaling, and failover, reducing the operational burden on the company's teams. It also supports various database engines and can easily integrate with Amazon Redshift, making it a flexible choice for this scenario.

In contrast, the other options involve more complexity or are not suitable for real-time data capture:

- Option A (AWS Glue): AWS Glue is more suited for batch ETL operations and wouldn't provide the near real-time data updates required by the company. - Option C (Amazon AppFlow SDK): Building a custom connector would involve significant development effort and time, which contradicts the requirement for minimal development effort.

- Option D (AWS DataSync): AWS DataSync is designed for transferring file-based data and not for real-time database replication, making it unsuitable for this scenario.

Therefore, using AWS DMS with a full load and CDC task is the most efficient and least development-intensive solution for continuously replicating MySQL database changes to Amazon Redshift.

Question # 106

Date: June 29, 2024

A marketing company uses Amazon S3 to store clickstream data. The company queries the data at the end of each day by using a SQL JOIN clause on S3 objects that are stored in separate buckets.

The company creates key performance indicators (KPIs) based on the objects. The company needs a serverless solution that will give users the ability to query data by partitioning the data. The solution must maintain the atomicity, consistency, isolation, and durability (ACID) properties of the data.

Which solution will meet these requirements MOST cost-effectively?

A. Amazon S3 Select
B. Amazon Redshift Spectrum
C. Amazon Athena
D. Amazon EMR

Correct Answer:

A. Amazon S3 Select

B. Amazon Redshift Spectrum

C. Amazon Athena

D. Amazon EMR

Explanation:

Amazon Athena is the correct choice for this scenario, and here's why:

1. Serverless Architecture: Amazon Athena is a serverless query service, which means you don't have to manage any infrastructure. This aligns perfectly with the requirement for a serverless solution. Users can simply execute queries on the data stored in Amazon S3 without needing to set up and manage any physical or virtual servers.

2. SQL Queries with JOINs: Athena allows you to run SQL queries, including complex operations like JOINs, directly on data stored in S3. This is essential for the marketing company that needs to perform SQL JOIN operations on clickstream data stored across different S3 objects in separate buckets.

3. Cost-effectiveness: Athena charges you based on the amount of data scanned by your queries. If you design your queries efficiently and partition your data appropriately, you can significantly reduce costs. This makes it a cost-effective solution compared to other options that might require maintaining and paying for a persistent cluster.

4. Data Partitioning: Athena supports data partitioning, which means you can organize your data in a way that allows for faster query performance and less data scanning. This is crucial for efficient data retrieval and cost savings, as it reduces the amount of data that Athena needs to scan when running queries.

5. ACID Properties: Although Athena itself does not fully guarantee ACID properties (as it is typically used for running analytical queries rather than transactional ones), it can work with data that is managed by AWS Glue, which can provide some level of data consistency and durability. For analytical queries like those described in the scenario, Athena is suitable as it can handle read-consistent data, particularly if data is well-partitioned and managed.

Comparing with other options:

- Amazon S3 Select is designed for retrieving subsets of object data from within Amazon S3, but it is not suited for running complex SQL queries with JOIN operations across multiple objects or buckets.

- Amazon Redshift Spectrum allows querying data in S3 using Redshift, but it involves setting up a Redshift cluster, which adds complexity and cost, especially if you need Redshift's full capabilities.

- Amazon EMR is a managed Hadoop framework that can perform big data processing tasks. While it is powerful, it involves managing clusters, which can be costlier and more complex than a serverless solution like Athena, especially for daily querying tasks.

In summary, Amazon Athena is the most suitable option for the company's needs due to its serverless nature, ability to perform SQL queries including JOINs, support for data partitioning, and cost-effectiveness.

Question # 107

Date: June 29, 2024

A company wants to migrate data from an Amazon RDS for PostgreSQL DB instance in the eu-east-1 Region of an AWS account named Account_A. The company will migrate the data to an Amazon Redshift cluster in the eu-west-1 Region of an AWS account named Account_B.

Which solution will give AWS Database Migration Service (AWS DMS) the ability to replicate data between two data stores?

A. Set up an AWS DMS replication instance in Account_B in eu-west-1.
B. Set up an AWS DMS replication instance in Account_B in eu-east-1.
C. Set up an AWS DMS replication instance in a new AWS account in eu-west-1.
D. Set up an AWS DMS replication instance in Account_A in eu-east-1.

Correct Answer:

A. Set up an AWS DMS replication instance in Account_B in eu-west-1.

B. Set up an AWS DMS replication instance in Account_B in eu-east-1.

C. Set up an AWS DMS replication instance in a new AWS account in eu-west-1.

D. Set up an AWS DMS replication instance in Account_A in eu-east-1.

Explanation:

In this scenario, the company wants to migrate data from an Amazon RDS for PostgreSQL instance in one AWS account (Account_A) located in the eu-east-1 region to an Amazon Redshift cluster in another AWS account (Account_B) located in the eu-west-1 region. The task involves cross-region and cross-account data replication, which can be effectively managed using AWS Database Migration Service (AWS DMS).

Here's why option A is the correct solution:

1. Location of the Target Database: The target database, Amazon Redshift, is located in the eu-west-1 region. Setting up the AWS DMS replication instance close to the target data store can help in reducing latency and potentially improving the performance of the data migration process. Thus, placing the DMS instance in the eu-west-1 region aligns with best practices for data migration, as it minimizes the distance over which the data has to travel during replication.

2. Account Ownership: Since the target database resides in Account_B, it makes sense to set up the DMS replication instance in the same account (Account_B). This strategy simplifies the process of granting necessary permissions and managing the data migration process. Managing resources within the same AWS account reduces complexity related to cross-account permissions and security.

3. Cross-Region and Cross-Account Migration: AWS DMS is designed to handle cross-region and cross-account migrations. By placing the replication instance in Account_B in the target region (eu-west-1), you ensure that the replication instance is closer to the target Redshift cluster, which is the final destination for the migrated data.

4. Network and Security Considerations: By deploying the replication instance in the same region and account as the target, you can efficiently manage network traffic and security configurations. This setup helps in configuring security groups, IAM roles, and other necessary configurations to allow the DMS instance to connect to both the source and target databases securely.

In summary, option A provides an optimal solution that aligns with best practices for AWS DMS, considering factors such as reducing latency, simplifying the management of permissions, and ensuring efficient and secure data transfer between the source and target databases.

Question # 108

Date: June 29, 2024

A company uses Amazon S3 as a data lake. The company sets up a data warehouse by using a multi-node Amazon Redshift cluster. The company organizes the data files in the data lake based on the data source of each data file.

The company loads all the data files into one table in the Redshift cluster by using a separate COPY command for each data file location. This approach takes a long time to load all the data files into the table. The company must increase the speed of the data ingestion. The company does not want to increase the cost of the process.

Which solution will meet these requirements?

A. Use a provisioned Amazon EMR cluster to copy all the data files into one folder. Use a COPY command to load the data into Amazon Redshift.
B. Load all the data files in parallel into Amazon Aurora. Run an AWS Glue job to load the data into Amazon Redshift.
C. Use an AWS Give job to copy all the data files into one folder. Use a COPY command to load the data into Amazon Redshift.
D. Create a manifest file that contains the data file locations. Use a COPY command to load the data into Amazon Redshift.

Correct Answer:

A. Use a provisioned Amazon EMR cluster to copy all the data files into one folder. Use a COPY command to load the data into Amazon Redshift.

B. Load all the data files in parallel into Amazon Aurora. Run an AWS Glue job to load the data into Amazon Redshift.

C. Use an AWS Give job to copy all the data files into one folder. Use a COPY command to load the data into Amazon Redshift.

D. Create a manifest file that contains the data file locations. Use a COPY command to load the data into Amazon Redshift.

Explanation:

The correct answer to the question is D: Create a manifest file that contains the data file locations. Use a COPY command to load the data into Amazon Redshift.

Explanation:

1. Understanding the Problem: - The company is using Amazon S3 as a data lake and wants to load this data into an Amazon Redshift cluster. - The data files are organized by data source, and each file is loaded into one table using separate COPY commands. This process is slow and inefficient.

2. Challenges with the Current Approach: - Executing a separate COPY command for each data file location is time-consuming because it doesn't leverage parallel processing capabilities. - The goal is to speed up data ingestion without increasing costs.

3. Why Option D is Correct: - Manifest File: A manifest file is a JSON file that lists all the S3 objects (data files) you want to load into Amazon Redshift. It provides a way to specify multiple files as input for a single COPY command. - Efficiency: By using a manifest file, you can execute a single COPY command that references all the data files listed in the manifest. This approach allows Redshift to load data in parallel from multiple files, significantly speeding up the data ingestion process. - Cost-Effective: This method uses existing AWS features without introducing additional services or resources that would increase costs.

4. Comparison with Other Options: - Option A (EMR Cluster): Using an EMR cluster to consolidate files into one folder introduces additional infrastructure and potential cost and complexity. It is not necessary just to organize files. - Option B (Amazon Aurora and AWS Glue): Loading data into Amazon Aurora first and then into Redshift via AWS Glue adds unnecessary complexity and cost. It involves additional data movement and requires maintaining two separate databases. - Option C (AWS Glue Job): Using AWS Glue to move files could simplify file organization but doesn't directly address the need for faster data loading into Redshift. It also incurs additional costs for the Glue service.

In summary, using a manifest file allows the company to leverage Amazon Redshift's parallel data loading capabilities efficiently and cost-effectively. This approach directly addresses the problem of slow data ingestion without requiring additional services or infrastructure.

Question # 109

Date: June 29, 2024

A company plans to use Amazon Kinesis Data Firehose to store data in Amazon S3. The source data consists of 2 MB .csv files. The company must convert the .csv files to JSON format. The company must store the files in Apache Parquet format.

Which solution will meet these requirements with the LEAST development effort?

A. Use Kinesis Data Firehose to convert the .csv files to JSON. Use an AWS Lambda function to store the files in Parquet format.
B. Use Kinesis Data Firehose to convert the .csv files to JSON and to store the files in Parquet format.
C. Use Kinesis Data Firehose to invoke an AWS Lambda function that transforms the .csv files to JSON and stores the files in Parquet format.
D. Use Kinesis Data Firehose to invoke an AWS Lambda function that transforms the .csv files to JSON. Use Kinesis Data Firehose to store the files in Parquet format.

Correct Answer:

A. Use Kinesis Data Firehose to convert the .csv files to JSON. Use an AWS Lambda function to store the files in Parquet format.

B. Use Kinesis Data Firehose to convert the .csv files to JSON and to store the files in Parquet format.

C. Use Kinesis Data Firehose to invoke an AWS Lambda function that transforms the .csv files to JSON and stores the files in Parquet format.

D. Use Kinesis Data Firehose to invoke an AWS Lambda function that transforms the .csv files to JSON. Use Kinesis Data Firehose to store the files in Parquet format.

Explanation:

The scenario involves using Amazon Kinesis Data Firehose to process and store data in a specified format with minimal development effort. Let's break down the requirements and how Option D addresses them effectively:

1. Data Transformation from CSV to JSON: - The company needs to convert CSV files into JSON format. Kinesis Data Firehose can utilize AWS Lambda for record processing. By configuring a Lambda function, you can perform custom transformations, such as converting CSV data into JSON. This allows you to leverage Lambda's flexibility to handle the conversion logic, which can be coded to handle various transformations as required.

2. Storing Data in Parquet Format: - After converting the data into JSON, the next requirement is to store it in Apache Parquet format. Kinesis Data Firehose natively supports converting data into formats like Parquet and ORC before delivery to Amazon S3. This feature can be enabled in the Firehose configuration without additional coding, making it a straightforward solution for converting JSON data into Parquet format.

3. Minimizing Development Effort: - The key is to achieve these transformations with the least development effort. Option D suggests using Kinesis Data Firehose in conjunction with AWS Lambda for the CSV to JSON conversion and then leveraging Firehose's built-in capability to convert JSON to Parquet. This approach minimizes custom development because: - The Lambda function only needs to handle the CSV to JSON transformation logic. - The built-in functionality of Firehose handles the JSON to Parquet conversion, eliminating the need to write custom code for this step.

In contrast: - Option A requires the Lambda function to handle both the format conversion and storage in Parquet, increasing complexity and development effort. - Option B assumes Firehose can directly convert CSV to JSON and then to Parquet, which is not supported without an intermediary transformation step. - Option C suggests using Lambda for both transformations, which also increases the development burden compared to leveraging Firehose's native capabilities for Parquet conversion.

Thus, Option D efficiently splits the tasks between Lambda and Firehose, making it the most efficient and least development-intensive solution.

Question # 110

Date: June 29, 2024

A company is using an AWS Transfer Family server to migrate data from an on-premises environment to AWS. Company policy mandates the use of TLS 1.2 or above to encrypt the data in transit.

Which solution will meet these requirements?

A. Generate new SSH keys for the Transfer Family server. Make the old keys and the new keys available for use.
B. Update the security group rules for the on-premises network to allow only connections that use TLS 1.2 or above.
C. Update the security policy of the Transfer Family server to specify a minimum protocol version of TLS 1.2
D. Install an SSL certificate on the Transfer Family server to encrypt data transfers by using TLS 1.2.

Correct Answer:

A. Generate new SSH keys for the Transfer Family server. Make the old keys and the new keys available for use.

B. Update the security group rules for the on-premises network to allow only connections that use TLS 1.2 or above.

C. Update the security policy of the Transfer Family server to specify a minimum protocol version of TLS 1.2

D. Install an SSL certificate on the Transfer Family server to encrypt data transfers by using TLS 1.2.

Explanation:

The correct answer is C: Update the security policy of the Transfer Family server to specify a minimum protocol version of TLS 1.2.

Here's why this is the correct solution:

AWS Transfer Family is a fully managed service that enables the transfer of files over SFTP, FTPS, and FTP directly into and out of Amazon S3. When dealing with data transfers, especially over the internet, securing the data in transit is crucial to prevent interception or tampering. Transport Layer Security (TLS) is a protocol that provides privacy and data integrity between two communicating applications.

The company policy requires the use of TLS 1.2 or above to secure data in transit. TLS 1.2 is a well-established standard that offers robust encryption and is commonly mandated by organizations for secure communications.

Option C is the correct answer because it involves updating the security policy of the AWS Transfer Family server to enforce a minimum protocol version of TLS 1.2. This configuration ensures that any client connecting to the server must use TLS 1.2 or a more recent version of the protocol, aligning with the company's security policy. By specifying a minimum protocol version, the server can reject older, less secure versions of TLS, thereby enhancing the security of data in transit.

Let's briefly consider why the other options are not suitable:

- Option A: Generating new SSH keys is related to securing SFTP connections through SSH, not TLS. This does not address the requirement to use TLS 1.2 for data encryption.

- Option B: Updating security group rules on the on-premises network is not a direct solution for enforcing TLS 1.2 on the Transfer Family server. Security groups control inbound and outbound traffic based on IP addresses and ports, not the protocol version used.

- Option D: Installing an SSL certificate on the Transfer Family server could help in establishing TLS connections, but it does not enforce the use of a specific minimum version of TLS. The certificate itself does not dictate the TLS version; the server's security policy does.

Therefore, the most direct and effective way to adhere to the company's policy is by updating the security policy of the server to enforce TLS 1.2 or higher, as described in option C.

Question # 111

Date: June 29, 2024

A company wants to migrate an application and an on-premises Apache Kafka server to AWS. The application processes incremental updates that an on-premises Oracle database sends to the Kafka server. The company wants to use the replatform migration strategy instead of the refactor strategy.

Which solution will meet these requirements with the LEAST management overhead?

A. Amazon Kinesis Data Streams
B. Amazon Managed Streaming for Apache Kafka (Amazon MSK) provisioned cluster
C. Amazon Kinesis Data Firehose
D. Amazon Managed Streaming for Apache Kafka (Amazon MSK) Serverless

Correct Answer:

A. Amazon Kinesis Data Streams

B. Amazon Managed Streaming for Apache Kafka (Amazon MSK) provisioned cluster

C. Amazon Kinesis Data Firehose

D. Amazon Managed Streaming for Apache Kafka (Amazon MSK) Serverless

Explanation:

The question involves migrating an application and an on-premises Apache Kafka server to AWS, with the requirement to use a replatform migration strategy, which involves moving to a different platform without significant changes to the application's code or architecture. The company also wants to minimize management overhead.

Let's break down why option D, Amazon Managed Streaming for Apache Kafka (Amazon MSK) Serverless, is the correct choice:

1. Replatform Strategy: This strategy focuses on moving to a new infrastructure environment without making significant changes to the application. Using Amazon MSK, which is essentially a managed Kafka service, aligns perfectly with this approach as it allows you to run Kafka on AWS with minimal changes to the existing Kafka-based application.

2. Least Management Overhead: Amazon MSK Serverless is designed to provide a Kafka environment without the need to manage the underlying infrastructure. This means AWS takes care of provisioning, scaling, and managing the Kafka clusters. Thus, it offers significant reductions in management tasks compared to a provisioned cluster or running Kafka on your own.

3. Compatibility with Existing Architecture: Since the application is already using Apache Kafka, migrating to Amazon MSK ensures compatibility with the existing Kafka-based application architecture. This makes the transition smoother and less error-prone compared to introducing a completely different service like Kinesis.

4. Serverless Benefits: By choosing the serverless option, the company benefits from automatic scaling based on demand, which eliminates the need to worry about scaling infrastructure manually or over-provisioning resources. This is ideal for workloads with variable or unpredictable traffic patterns.

5. Cost Efficiency: With MSK Serverless, the company pays only for the data they stream and consume, which can be more cost-effective compared to managing dedicated Kafka clusters, as it avoids costs associated with over-provisioning resources.

In summary, Amazon MSK Serverless provides the most seamless transition for an existing Kafka-based system while minimizing management and operational overhead, aligning perfectly with the company's requirements to migrate with the least management burden.

Question # 112

Date: June 29, 2024

A data engineer is building an automated extract, transform, and load (ETL) ingestion pipeline by using AWS Glue. The pipeline ingests compressed files that are in an Amazon S3 bucket. The ingestion pipeline must support incremental data processing.

Which AWS Glue feature should the data engineer use to meet this requirement?

A. Workflows
B. Triggers

C. Job bookmarks
D. Classifiers

Correct Answer:

A. Workflows

B. Triggers

C. Job bookmarks

D. Classifiers

Explanation:
The correct answer is C: Job bookmarks.

When building an ETL pipeline using AWS Glue, especially one that needs to support incremental data processing, job bookmarks are an essential feature. Here's why:

1. Incremental Data Processing: Incremental processing means that the data pipeline only processes new or updated data rather than reprocessing the entire dataset each time the job runs. This approach is more efficient and reduces the overall processing time and cost.

2. Function of Job Bookmarks: Job bookmarks in AWS Glue are designed to keep track of previously processed data. They essentially remember the state of data processing from the last successful run of a Glue job. This means that when the job runs again, it can pick up from where it left off, processing only the new data that has arrived since the last run.

3. Efficiency and Cost-Effectiveness: By using job bookmarks, you avoid the need to reprocess all the data, which not only saves time but also reduces the computational resources required, leading to cost savings.

4. Use Case Alignment: In the scenario described, the data engineer is dealing with compressed files in an Amazon S3 bucket. As new data arrives or existing data is updated, job bookmarks allow the pipeline to process only these changes, ensuring that the ETL process is both efficient and up-to-date.

5. How Job Bookmarks Work: When a Glue job with job bookmarks enabled runs, it writes metadata about the processed data to the Glue Data Catalog. On subsequent runs, it uses this metadata to determine which data has already been processed and to skip over it, focusing only on new or modified data.

In summary, job bookmarks are specifically designed for managing state and ensuring that only new or updated data is processed during ETL jobs, making them the ideal choice for supporting incremental data processing in an AWS Glue pipeline. Options A (Workflows), B (Triggers), and D (Classifiers) do not provide this functionality of state management and incremental data processing, which is why they are not the correct choices in this context.

Question # 113

Date: August 03, 2024

A company stores petabytes of data in thousands of Amazon S3 buckets in the S3 Standard storage class. The data supports analytics workloads that have unpredictable and variable data access patterns.

The company does not access some data for months. However, the company must be able to retrieve all data within milliseconds. The company needs to optimize S3 storage costs.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost-optimized storage classes. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.
B. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequently. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.
C. Use S3 Intelligent-Tiering. Activate the Deep Archive Access tier.
D. Use S3 Intelligent-Tiering. Use the default access tier.

Correct Answer:

A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost-optimized storage classes. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.

B. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequently. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.

C. Use S3 Intelligent-Tiering. Activate the Deep Archive Access tier.

D. Use S3 Intelligent-Tiering. Use the default access tier.

Explanation:

The correct answer is D: Use S3 Intelligent-Tiering. Use the default access tier.

Here's why this is the best solution:

1. Unpredictable and Variable Access Patterns: The company needs to manage data with unpredictable and variable access patterns. This means that at times the data might be accessed frequently, while at other times, it might not be accessed at all for months. S3 Intelligent-Tiering is specifically designed to handle such scenarios without requiring manual intervention.

2. Cost Optimization: S3 Intelligent-Tiering automatically moves data between two access tiers: a frequent access tier and an infrequent access tier, based on changing access patterns. This allows the company to optimize costs automatically because it charges a lower rate for data in the infrequent access tier, without compromising on retrieval times.

3. Retrieval within Milliseconds: The requirement to retrieve all data within milliseconds means that the data cannot be moved to cold storage options like S3 Glacier or S3 Glacier Deep Archive, as these storage classes have retrieval times ranging from minutes to hours. S3 Intelligent-Tiering, on the other hand, keeps all the data immediately accessible, regardless of the access tier, meeting the requirement for millisecond retrieval.

4. Least Operational Overhead: The company seeks a solution with minimal operational overhead. S3 Intelligent-Tiering requires no lifecycle policies or manual monitoring to move data between storage classes. Once enabled, it automatically adjusts the data's storage class based on actual access patterns, eliminating the need for ongoing management and manual intervention.

5. Default Access Tier: Using the default access tier configuration of S3 Intelligent-Tiering ensures that the setup is straightforward and doesn't require additional configuration steps, which further reduces operational overhead.

In summary, S3 Intelligent-Tiering provides a fully automated, cost-effective solution that optimizes storage costs while ensuring immediate data availability, perfectly aligning with the company's requirements without adding complexity or management burden.

Question # 114

Date: June 30, 2024

A company stores logs in an Amazon S3 bucket. When a data engineer attempts to access several log files, the data engineer discovers that some files have been unintentionally deleted.

The data engineer needs a solution that will prevent unintentional file deletion in the future.

Which solution will meet this requirement with the LEAST operational overhead?

A. Manually back up the S3 bucket on a regular basis.
B. Enable S3 Versioning for the S3 bucket.

C. Configure replication for the S3 bucket.
D. Use an Amazon S3 Glacier storage class to archive the data that is in the S3 bucket.

Correct Answer:

A. Manually back up the S3 bucket on a regular basis.

B. Enable S3 Versioning for the S3 bucket.

C. Configure replication for the S3 bucket.

D. Use an Amazon S3 Glacier storage class to archive the data that is in the S3 bucket.

Explanation:

The correct answer is B: Enable S3 Versioning for the S3 bucket.

Here's why this is the best solution with the least operational overhead:

1. Understanding S3 Versioning: Enabling S3 Versioning on an Amazon S3 bucket allows you to keep multiple versions of an object in the same bucket. This means that every time an object is modified or deleted, the previous version is retained. This feature is particularly useful for protecting against accidental deletions or overwrites, as it allows you to easily restore an earlier version of an object if needed.

2. Protection Against Unintentional Deletion: With versioning enabled, if a file is unintentionally deleted, it isn't permanently removed. Instead, it's marked as a "delete marker" while previous versions of the file still exist in the bucket. This allows the data engineer to recover the file by simply removing the delete marker or reverting to an earlier version.

3. Minimal Operational Overhead: Enabling S3 Versioning is a straightforward process that requires just a one-time configuration change to the bucket settings. Once enabled, it operates automatically without requiring manual intervention or complex setup. This means there's no need for ongoing manual processes or additional management tasks, which keeps operational overhead low.

4. Cost Considerations: While enabling versioning might increase storage costs because you're storing multiple versions of the same object, it does not require additional resources or complex applications to manage. The benefit of easily recovering files generally outweighs the potential for slightly higher storage costs.

5. Comparing Other Options: - Option A (Manually back up the S3 bucket regularly): This requires ongoing manual effort to create and manage backups, which introduces significant operational overhead. - Option C (Configure replication for the S3 bucket): Replication is primarily used for disaster recovery and compliance by creating copies of data in different regions or buckets. While it protects against regional failures, it doesn't inherently protect against accidental deletions as the deletion would be replicated as well. - Option D (Use Amazon S3 Glacier): Glacier is a storage class for archival purposes, designed for long-term storage of infrequently accessed data. It doesn't prevent deletion and involves additional steps to retrieve data, which can add complexity and latency.

In summary, enabling S3 Versioning is a simple, effective, and low-maintenance way to safeguard your data against accidental deletions, making it the ideal choice for this scenario with the least operational overhead.

Question # 115

Date: June 30, 2024

A telecommunications company collects network usage data throughout each day at a rate of several thousand data points each second. The company runs an application to process the usage data in real time. The company aggregates and stores the data in an Amazon Aurora DB instance.

Sudden drops in network usage usually indicate a network outage. The company must be able to identify sudden drops in network usage so the company can take immediate remedial actions.

Which solution will meet this requirement with the LEAST latency?

A. Create an AWS Lambda function to query Aurora for drops in network usage. Use Amazon EventBridge to automatically invoke the Lambda function every minute.
B. Modify the processing application to publish the data to an Amazon Kinesis data stream. Create an Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) application to detect drops in network usage.
C. Replace the Aurora database with an Amazon DynamoDB table. Create an AWS Lambda function to query the DynamoDB table for drops in network usage every minute. Use DynamoDB Accelerator (DAX) between the processing application and DynamoDB table.
D. Create an AWS Lambda function within the Database Activity Streams feature of Aurora to detect drops in network usage.

Correct Answer:

A. Create an AWS Lambda function to query Aurora for drops in network usage. Use Amazon EventBridge to automatically invoke the Lambda function every minute.

B. Modify the processing application to publish the data to an Amazon Kinesis data stream. Create an Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) application to detect drops in network usage.

C. Replace the Aurora database with an Amazon DynamoDB table. Create an AWS Lambda function to query the DynamoDB table for drops in network usage every minute. Use DynamoDB Accelerator (DAX) between the processing application and DynamoDB table.

D. Create an AWS Lambda function within the Database Activity Streams feature of Aurora to detect drops in network usage.

Explanation:

The correct answer is B. Let's explore why this solution is the best choice for detecting sudden drops in network usage with the least latency:

1. Real-time Data Processing: The requirement is to identify sudden drops in network usage in real-time. Amazon Kinesis is designed for high-throughput data streaming and real-time analytics. By modifying the processing application to publish data to an Amazon Kinesis data stream, the company can process and analyze data as it arrives.

2. Amazon Managed Service for Apache Flink: Formerly known as Amazon Kinesis Data Analytics, this service allows for real-time stream processing and analytics. It can continuously process and analyze streaming data with minimal latency. Apache Flink is well-suited for detecting patterns or anomalies, like sudden drops in usage, as the data flows through the stream.

3. Low Latency: This solution provides low latency because it processes data in real-time as it streams in. Unlike batch processing or querying a database periodically, stream processing with Apache Flink offers near-instantaneous insight into data trends and anomalies.

4. Scalability: Kinesis data streams can handle massive volumes of data, which is crucial given that the telecommunications company collects several thousand data points each second. This ensures scalability and reliability under high data throughput scenarios.

5. Seamless Integration: AWS services like Kinesis and Flink are designed to work well together, providing a seamless and efficient workflow for real-time data processing. This integration helps in quickly detecting and responding to network usage drops.

In contrast, other options involve querying a database periodically or relying on batch processing, which introduces higher latency and delayed detection of network issues. For instance, using Lambda functions to query databases every minute (as in options A and C) introduces significant delays in detecting sudden changes, making them less suitable for real-time needs. Option D, while leveraging Aurora's features, still involves a database-centric approach, which is not as efficient as stream processing for real-time detection.

Overall, option B leverages the strengths of real-time streaming and analytics services offered by AWS to meet the company's needs with the least latency.

Question # 116

Date: August 07, 2024

A data engineer is processing and analyzing multiple terabytes of raw data that is in Amazon S3. The data engineer needs to clean and prepare the data. Then the data engineer needs to load the data into Amazon Redshift for analytics.

The data engineer needs a solution that will give data analysts the ability to perform complex queries. The solution must eliminate the need to perform complex extract, transform, and load (ETL) processes or to manage infrastructure.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon EMR to prepare the data. Use AWS Step Functions to load the data into Amazon Redshift. Use Amazon QuickSight to run queries.
B. Use AWS Glue DataBrew to prepare the data. Use AWS Glue to load the data into Amazon Redshift. Use Amazon Redshift to run queries.
C. Use AWS Lambda to prepare the data. Use Amazon Kinesis Data Firehose to load the data into Amazon Redshift. Use Amazon Athena to run queries.
D. Use AWS Glue to prepare the data. Use AWS Database Migration Service (AVVS DMS) to load the data into Amazon Redshift. Use Amazon Redshift Spectrum to run queries.

Correct Answer:

A. Use Amazon EMR to prepare the data. Use AWS Step Functions to load the data into Amazon Redshift. Use Amazon QuickSight to run queries.

B. Use AWS Glue DataBrew to prepare the data. Use AWS Glue to load the data into Amazon Redshift. Use Amazon Redshift to run queries.

C. Use AWS Lambda to prepare the data. Use Amazon Kinesis Data Firehose to load the data into Amazon Redshift. Use Amazon Athena to run queries.

D. Use AWS Glue to prepare the data. Use AWS Database Migration Service (AVVS DMS) to load the data into Amazon Redshift. Use Amazon Redshift Spectrum to run queries.

Explanation:

The correct answer is B, which suggests using AWS Glue DataBrew to prepare the data, using AWS Glue to load the data into Amazon Redshift, and using Amazon Redshift to run queries. Let's break down why this solution is ideal for the scenario described:

1. AWS Glue DataBrew: This is a visual data preparation tool that allows you to clean and normalize data without writing code. It is designed for data engineers and analysts to easily handle large datasets, such as multiple terabytes of raw data stored in Amazon S3. Since it provides a no-code interface, it significantly reduces the complexity and operational overhead compared to traditional, manual ETL processes.

2. AWS Glue: A fully managed ETL service that makes it easy to prepare and load data for analytics. AWS Glue can seamlessly integrate with Amazon Redshift, allowing you to automate the process of loading data into Redshift. This integration reduces the need for manual infrastructure management and complex ETL code, thus meeting the requirement for minimal operational overhead.

3. Amazon Redshift: A powerful, fully managed data warehouse service designed to handle large-scale data analytics. By loading data into Amazon Redshift, you provide data analysts with the ability to perform complex queries efficiently. Redshift is optimized for running complex analytical queries on large datasets, which fits well with the requirements of the scenario.

By combining AWS Glue DataBrew and AWS Glue, the data engineer can streamline the preparation and loading phases without managing the underlying infrastructure. This approach not only simplifies the process but also ensures scalability and efficiency. Using Amazon Redshift as the querying engine further enhances the capability to perform complex analytics on large datasets. Overall, this solution fulfills the requirement for the least operational overhead while providing robust data analytics capabilities.

Question # 117

Date: August 07, 2024

A company uses an AWS Lambda function to transfer files from a legacy SFTP environment to Amazon S3 buckets. The Lambda function is VPC enabled to ensure that all communications between the Lambda function and other AVS services that are in the same VPC environment will occur over a secure network.

The Lambda function is able to connect to the SFTP environment successfully. However, when the Lambda function attempts to upload files to the S3 buckets, the Lambda function returns timeout errors. A data engineer must resolve the timeout issues in a secure way.

Which solution will meet these requirements in the MOST cost-effective way?

A. Create a NAT gateway in the public subnet of the VPC. Route network traffic to the NAT gateway.
B. Create a VPC gateway endpoint for Amazon S3. Route network traffic to the VPC gateway endpoint.
C. Create a VPC interface endpoint for Amazon S3. Route network traffic to the VPC interface endpoint.
D. Use a VPC internet gateway to connect to the internet. Route network traffic to the VPC internet gateway.

Correct Answer:

A. Create a NAT gateway in the public subnet of the VPC. Route network traffic to the NAT gateway.

B. Create a VPC gateway endpoint for Amazon S3. Route network traffic to the VPC gateway endpoint.

C. Create a VPC interface endpoint for Amazon S3. Route network traffic to the VPC interface endpoint.

D. Use a VPC internet gateway to connect to the internet. Route network traffic to the VPC internet gateway.

Explanation:

The problem described involves a Lambda function that is VPC-enabled and needs to upload files to Amazon S3 but is experiencing timeout errors. The core issue here is that the Lambda function, being within a VPC, does not have direct internet access to reach S3. By default, resources within a VPC do not have internet access unless explicitly provided, which could lead to these timeout errors when attempting to access S3.

Let's break down why option B, "Create a VPC gateway endpoint for Amazon S3. Route network traffic to the VPC gateway endpoint," is the most cost-effective and suitable solution:

1. Understanding VPC Endpoints: - VPC endpoints allow connectivity between a VPC and AWS services without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect. They are virtual devices that enable you to privately connect your VPC to supported AWS services.

2. Gateway vs. Interface Endpoints: - Gateway Endpoints are used for services like Amazon S3 and DynamoDB. They are highly scalable and redundant VPC components that require no additional cost beyond data transfer charges, similar to accessing S3 directly over the internet. - Interface Endpoints are used for most other AWS services and involve the use of AWS PrivateLink, which can incur additional costs.

3. Cost Consideration: - A VPC gateway endpoint for S3 does not incur extra charges beyond the typical S3 data transfer costs, making it cost-effective. - NAT gateways, on the other hand, incur additional hourly and data processing charges, which can add up quickly, especially for data-heavy operations.

4. Security and Simplicity: - Using a VPC gateway endpoint allows the Lambda function to access S3 directly within the AWS network without exposing traffic to the public internet, enhancing security. - It simplifies network architecture by eliminating the need for public IP addresses, internet gateways, or NAT instances/gateways, reducing complexity and points of failure.

5. Routing Traffic: - By creating a VPC endpoint for S3, traffic between the Lambda function and S3 is routed internally within AWS's network, which is efficient and reliable.

In summary, option B is the best choice because it provides a secure, cost-effective, and straightforward way to connect the Lambda function to S3 without needing internet access. It is tailored for scenarios like this, where internal AWS service communication is required from within a VPC.

Question # 118

Date: August 06, 2024

A company uploads .csv files to an Amazon S3 bucket. The company's data platform team has set up an AWS Glue crawler to perform data discovery and to create the tables and schemas.

An AWS Glue job writes processed data from the tables to an Amazon Redshift database. The AWS Glue job handles column mapping and creates the Amazon Redshift tables in the Redshift database appropriately.

If the company reruns the AWS Glue job for any reason, duplicate records are introduced into the Amazon Redshift tables. The company needs a solution that will update the Redshift tables without duplicates.

Which solution will meet these requirements?

A. Modify the AWS Glue job to copy the rows into a staging Redshift table. Add SQL commands to update the existing rows with new values from the staging Redshift table.
B. Modify the AWS Glue job to load the previously inserted data into a MySQL database. Perform an upsert operation in the MySQL database. Copy the results to the Amazon Redshift tables.
C. Use Apache Spark's DataFrame dropDuplicates() API to eliminate duplicates. Write the data to the Redshift tables.
D. Use the AWS Glue ResolveChoice built-in transform to select the value of the column from the most recent record.

Correct Answer:

A. Modify the AWS Glue job to copy the rows into a staging Redshift table. Add SQL commands to update the existing rows with new values from the staging Redshift table.

B. Modify the AWS Glue job to load the previously inserted data into a MySQL database. Perform an upsert operation in the MySQL database. Copy the results to the Amazon Redshift tables.

C. Use Apache Spark's DataFrame dropDuplicates() API to eliminate duplicates. Write the data to the Redshift tables.

D. Use the AWS Glue ResolveChoice built-in transform to select the value of the column from the most recent record.

Explanation:

The correct answer is A. Here's why this solution meets the company's requirements:

When dealing with data ingestion into Amazon Redshift, especially when data is ingested multiple times, there's a risk of introducing duplicate records. This happens because each run of the AWS Glue job appends the data to the existing tables in Redshift without any checks for duplicates.

Option A suggests modifying the AWS Glue job to first copy the new rows into a staging table in Amazon Redshift. This is a common strategy to handle data deduplication and updates. By using a staging table, you can perform operations that compare new incoming data against existing data in the main Redshift table. The key here is using SQL commands to update existing rows in the main Redshift table with new values from the staging table.

Here's a breakdown of how this approach works:

1. Staging Table: When data is first copied to a staging table, it acts as a temporary holding area. This separation allows you to manipulate and clean the data without immediately affecting the production data.

2. Comparison and Updating: Once the data is in the staging table, you can run SQL commands to check for duplicates or changes. For instance, you could use SQL commands like `MERGE` or a combination of `UPDATE` and `INSERT` to ensure that only new or updated records are added to the main table. This process is often referred to as an "upsert" (update or insert).

3. No Duplicates: By using this method, you ensure that the main Redshift table only contains the most recent data without duplicates. If a row already exists, it can be updated with new values from the staging table rather than being inserted again.

In contrast, the other options do not effectively address the problem of duplicates or are not optimal for the given scenario: - Option B introduces unnecessary complexity by involving a MySQL database, which is not needed for this task. - Option C suggests using Spark's `dropDuplicates()` function, which may not be sufficient if the duplicates arise from many sources or complex conditions that are better handled with SQL logic. - Option D talks about using AWS Glue’s `ResolveChoice` transform, which is generally used for resolving data type conflicts or column choice conflicts rather than deduplication.

Thus, option A is the most straightforward and effective way to ensure the Redshift tables are updated without introducing duplicates.

Question # 119 Date: August 09, 2024

A finance company receives data from third-party data providers and stores the data as objects in an Amazon S3 bucket.

The company ran an AWS Glue crawler on the objects to create a data catalog. The AWS Glue crawler created multiple tables. However, the company expected that the crawler would create only one table.

The company needs a solution that will ensure the AVS Glue crawler creates only one table.

Which combination of solutions will meet this requirement? (Choose two.)

A. Ensure that the object format, compression type, and schema are the same for each object.
B. Ensure that the object format and schema are the same for each object. Do not enforce consistency for the compression type of each object.
C. Ensure that the schema is the same for each object. Do not enforce consistency for the file format and compression type of each object.
D. Ensure that the structure of the prefix for each S3 object name is consistent.
E. Ensure that all S3 object names follow a similar pattern.

Correct Answer:

A. Ensure that the object format, compression type, and schema are the same for each object.

B. Ensure that the object format and schema are the same for each object. Do not enforce consistency for the compression type of each object.

C. Ensure that the schema is the same for each object. Do not enforce consistency for the file format and compression type of each object.

D. Ensure that the structure of the prefix for each S3 object name is consistent.

E. Ensure that all S3 object names follow a similar pattern.

Explanation:

Certainly! Let's break down why options A and D are the correct choices for ensuring that an AWS Glue crawler creates only one table when processing data stored as objects in an Amazon S3 bucket.

AWS Glue crawlers are designed to scan the data in your S3 buckets and automatically create tables in the AWS Glue Data Catalog based on the data's structure and format. The crawler does this by inferring the schema from the data it encounters. If the data varies significantly in format, schema, or organization, the crawler may create multiple tables, which is what the finance company encountered.

Here's why each selected option contributes to ensuring a single table is created:

A. Ensure that the object format, compression type, and schema are the same for each object.

- Object Format: The format (e.g., CSV, JSON, Parquet) must be consistent across all data objects. If different formats are used, the crawler will treat these as distinct datasets and create separate tables for each format.

- Compression Type: Compression affects how data is read. Using a consistent compression type ensures that the crawler can process all objects uniformly. If different compression types are used, the crawler may interpret them as different datasets.

- Schema: The schema, which includes the structure and types of data fields, must be identical for each object. If there are discrepancies in schemas across objects, the crawler will generate multiple tables to accommodate the different schemas.

By ensuring uniformity in these aspects, you help the crawler interpret all the data as part of a single dataset, leading to the creation of one table.

D. Ensure that the structure of the prefix for each S3 object name is consistent.

- Prefix Structure: The prefix in an S3 object name refers to the "folder-like" path before the actual object name. Consistent prefix structures help the crawler group objects logically. If prefixes vary significantly, the crawler might assume they belong to different datasets, leading to multiple tables.

- Consistent prefix structures also help in partitioning data within a single table, which is a common practice for optimizing query performance and management in data lakes.

In summary, ensuring consistency in object format, compression, schema, and the S3 object name prefix structure helps the AWS Glue crawler recognize all the data as belonging to a single logical dataset, thus creating only one table in the Data Catalog. This approach aligns with the company's requirement to consolidate the data representation into a single table.

Question # 120

Date: August 14, 2024

An application consumes messages from an Amazon Simple Queue Service (Amazon SQS) queue. The application experiences occasional downtime. As a result of the downtime, messages within the queue expire and are deleted after 1 day. The message deletions cause data loss for the application.

Which solutions will minimize data loss for the application? (Choose two.)

A. Increase the message retention period
B. Increase the visibility timeout.
C. Attach a dead-letter queue (DLQ) to the SQS queue.
D. Use a delay queue to delay message delivery
E. Reduce message processing time.

Correct Answer:

A. Increase the message retention period

B. Increase the visibility timeout.

C. Attach a dead-letter queue (DLQ) to the SQS queue.

D. Use a delay queue to delay message delivery

E. Reduce message processing time.

Explanation:

Certainly! Let's break down why options A and C are the correct choices for minimizing data loss in the scenario where an application consumes messages from an Amazon SQS queue and experiences occasional downtime.

Option A: Increase the message retention period

Increasing the message retention period is a direct solution to address the issue of messages expiring and getting deleted after 1 day. By default, Amazon SQS retains messages for up to 4 days, but this can be configured to retain messages for up to 14 days. By extending the retention period, you provide a larger window of time for the application to recover from downtime and process the messages before they are permanently deleted. This helps prevent data loss due to the expiration of messages that haven't been processed within the initial 1-day period.

Option C: Attach a dead-letter queue (DLQ) to the SQS queue

A dead-letter queue (DLQ) is a useful feature in Amazon SQS that allows you to capture messages that can't be processed successfully within a certain number of attempts. By attaching a DLQ to your main SQS queue, messages that fail processing can be redirected here instead of being lost. This provides an opportunity to inspect, reprocess, or analyze them later, thus significantly minimizing the risk of data loss. The DLQ acts as a safety net for messages that would otherwise be discarded due to repeated failures or application downtime.

Why the other options are not suitable:

- Option B: Increase the visibility timeout: This option is about controlling how long a message remains invisible to other consumers while it is being processed. While increasing the visibility timeout can help in avoiding duplicate processing of messages, it doesn't directly address message expiration or data loss due to application downtime.

- Option D: Use a delay queue to delay message delivery: A delay queue postpones the delivery of new messages to the queue. This can help in managing message flow, but it doesn't prevent existing messages from expiring if the application is down for extended periods. It also doesn't recover messages after they have already been deleted.

- Option E: Reduce message processing time: While reducing message processing time can help in processing messages faster, it doesn't address the problem of message expiration directly. It also doesn't provide a solution for when the application is completely down.

In summary, increasing the message retention period and attaching a dead-letter queue are both proactive measures that ensure messages are retained longer and have a backup mechanism in place, thereby minimizing the risk of data loss due to application downtime.

Question # 121

Date: August 14, 2024

A company is creating near real-time dashboards to visualize time series data. The company ingests data into Amazon Managed Streaming for Apache Kafka (Amazon MSK). A customized data pipeline consumes the data. The pipeline then writes data to Amazon Keyspaces (for Apache Cassandra), Amazon OpenSearch Service, and Apache Avro objects in Amazon S3.

Which solution will make the data available for the data visualizations with the LEAST latency?

A. Create OpenSearch Dashboards by using the data from OpenSearch Service.
B. Use Amazon Athena with an Apache Hive metastore to query the Avro objects in Amazon S3. Use Amazon Managed Grafana to connect to Athena and to create the dashboards.
C. Use Amazon Athena to query the data from the Avro objects in Amazon S3. Configure Amazon Keyspaces as the data catalog. Connect Amazon QuickSight to Athena to create the dashboards.
D. Use AWS Glue to catalog the data. Use S3 Select to query the Avro objects in Amazon S3. Connect Amazon QuickSight to the S3 bucket to create the dashboards.

Correct Answer:

A. Create OpenSearch Dashboards by using the data from OpenSearch Service.

B. Use Amazon Athena with an Apache Hive metastore to query the Avro objects in Amazon S3. Use Amazon Managed Grafana to connect to Athena and to create the dashboards.

C. Use Amazon Athena to query the data from the Avro objects in Amazon S3. Configure Amazon Keyspaces as the data catalog. Connect Amazon QuickSight to Athena to create the dashboards.

D. Use AWS Glue to catalog the data. Use S3 Select to query the Avro objects in Amazon S3. Connect Amazon QuickSight to the S3 bucket to create the dashboards.

Explanation:

The correct answer is A: Create OpenSearch Dashboards by using the data from OpenSearch Service.

Here's why this is the best solution for minimizing latency when creating near real-time dashboards:

1. Data Pipeline and OpenSearch Service: In the given scenario, data is being ingested into Amazon MSK and then processed by a custom data pipeline that writes data to Amazon Keyspaces, Amazon OpenSearch Service, and Apache Avro objects in Amazon S3. Among these, Amazon OpenSearch Service is specifically designed for indexing, searching, and analyzing large volumes of data quickly, which makes it ideal for real-time data visualization and analytics.

2. OpenSearch Dashboards: OpenSearch Dashboards is a tool specifically built to visualize data stored in Amazon OpenSearch Service. It provides real-time interactive data visualizations and dashboards, allowing users to create visual representations of data with minimal delay. This makes it an optimal choice for applications that require up-to-the-minute visualization of data, such as time series data in dashboards.

3. Low Latency: Since OpenSearch Dashboards directly accesses data from Amazon OpenSearch Service, it eliminates additional processing or querying layers that could introduce latency. This direct integration ensures that data visualizations are updated almost immediately as new data is ingested and indexed by OpenSearch Service.

4. Alternative Options and Latency: - Option B: Using Amazon Athena with an Apache Hive metastore to query Avro objects in Amazon S3 introduces additional querying time and complexity, which can add latency. Then connecting the results to Amazon Managed Grafana for visualization adds another layer, further increasing the time to visualize the data. - Option C: Similar to Option B, Athena is used to query Avro objects, which can be less efficient due to the time needed to execute the queries. Additionally, connecting through Amazon QuickSight adds more complexity and potential delay in rendering the data. - Option D: This option uses AWS Glue and S3 Select, which involves cataloging and querying data from S3, leading to higher latency compared to direct indexing and querying provided by OpenSearch Service and OpenSearch Dashboards.

In summary, Option A is the most efficient choice for achieving the least latency in this scenario because it uses OpenSearch Dashboards, which are natively integrated with Amazon OpenSearch Service, offering fast and direct access to the data for real-time visualizations.

Question # 122

Date: August 14, 2024

A media company wants to use Amazon OpenSearch Service to analyze rea-time data about popular musical artists and songs. The company expects to ingest millions of new data events every day. The new data events will arrive through an Amazon Kinesis data stream. The company must transform the data and then ingest the data into the OpenSearch Service domain.

Which method should the company use to ingest the data with the LEAST operational overhead?

A. Use Amazon Kinesis Data Firehose and an AWS Lambda function to transform the data and deliver the transformed data to OpenSearch Service.
B. Use a Logstash pipeline that has prebuilt filters to transform the data and deliver the transformed data to OpenSearch Service.
C. Use an AWS Lambda function to call the Amazon Kinesis Agent to transform the data and deliver the transformed data OpenSearch Service.
D. Use the Kinesis Client Library (KCL) to transform the data and deliver the transformed data to OpenSearch Service.

Correct Answer:

A. Use Amazon Kinesis Data Firehose and an AWS Lambda function to transform the data and deliver the transformed data to OpenSearch Service.

B. Use a Logstash pipeline that has prebuilt filters to transform the data and deliver the transformed data to OpenSearch Service.

C. Use an AWS Lambda function to call the Amazon Kinesis Agent to transform the data and deliver the transformed data OpenSearch Service.

D. Use the Kinesis Client Library (KCL) to transform the data and deliver the transformed data to OpenSearch Service.

Explanation:

The correct answer is A, and here's why:

The company needs to ingest data with the least operational overhead while transforming it and delivering it to Amazon OpenSearch Service. Let's break down the solution:

1. Amazon Kinesis Data Firehose: This is a fully managed service that makes it easy to load streaming data into data lakes, data stores, and analytics services. It automatically scales to match the throughput of your data and requires minimal setup and maintenance, which significantly reduces operational overhead. Kinesis Data Firehose can directly deliver data to Amazon OpenSearch Service, making it an excellent choice for this use case.

2. AWS Lambda for Data Transformation: Lambda is a serverless compute service that automatically scales and handles the infrastructure for you, allowing you to focus on the code that processes the data. By using a Lambda function, you can transform the data in real-time as it flows through the Kinesis Data Firehose stream. This integration is straightforward and doesn't require managing any servers or complex configurations.

3. Integration and Simplicity: The combination of Kinesis Data Firehose and AWS Lambda enables seamless integration and transformation of streaming data with minimal effort. You can set up the data pipeline quickly without needing to manage the underlying infrastructure, which aligns perfectly with the requirement of having the least operational overhead.

In contrast, the other options involve more complexity and operational effort:

- Option B (Logstash): While Logstash is capable of transforming and delivering data, it requires managing and configuring the Logstash instance, which increases operational overhead compared to the serverless and managed approach of Kinesis Data Firehose with Lambda.

- Option C (Kinesis Agent with Lambda): This approach is more complex than necessary. The Kinesis Agent is typically used for collecting and sending data to Kinesis streams from various sources, not transforming it. Using Lambda to call the agent adds unnecessary complexity.

- Option D (Kinesis Client Library): The KCL requires you to write and manage custom code to process the data, which involves more operational overhead than using the managed service of Kinesis Data Firehose. It also requires maintaining the application and infrastructure needed to run the KCL.

Therefore, using Amazon Kinesis Data Firehose with a Lambda function for transformation provides a streamlined, scalable, and managed solution with the least operational overhead, making option A the optimal choice.

Question # 123

Date: August 09, 2024

A company stores customer data tables that include customer addresses in an AWS Lake Formation data lake. To comply with new regulations, the company must ensure that users cannot access data for customers who are in Canada.

The company needs a solution that will prevent user access to rows for customers who are in Canada.

Which solution will meet this requirement with the LEAST operational effort?

A. Set a row-level filter to prevent user access to a row where the country is Canada.
B. Create an IAM role that restricts user access to an address where the country is Canada.
C. Set a column-level filter to prevent user access to a row where the country is Canada.
D. Apply a tag to all rows where Canada is the country. Prevent user access where the tag is equal to “Canada”.

Correct Answer:

A. Set a row-level filter to prevent user access to a row where the country is Canada.

B. Create an IAM role that restricts user access to an address where the country is Canada.

C. Set a column-level filter to prevent user access to a row where the country is Canada.

D. Apply a tag to all rows where Canada is the country. Prevent user access where the tag is equal to “Canada”.

Explanation:

The correct answer is A: Set a row-level filter to prevent user access to a row where the country is Canada.

Here's why this is the best solution with the least operational effort:

1. Row-Level Filtering: AWS Lake Formation provides the capability to implement fine-grained access controls on the data stored in your data lake. Row-level filtering is a feature that allows you to restrict access to specific rows in a table based on certain conditions, such as the values in one or more columns. By setting a row-level filter, you can specify that users should not be able to access any row where the country column is set to "Canada." This directly addresses the requirement to block access to data for Canadian customers.

2. Ease of Implementation: Implementing a row-level filter in AWS Lake Formation is relatively straightforward and involves setting up data lake permissions that define which data can be accessed under what conditions. This can be done through the Lake Formation console, API, or AWS CLI. Once set up, this filter automatically applies to all users and queries that attempt to access the data, ensuring compliance with minimal ongoing operational effort.

3. Granularity: Row-level filtering is more granular than column-level filtering or using tags. It specifically targets the rows of data you want to restrict access to, without affecting access to other parts of the data. This is crucial for cases where you want to maintain access to non-Canadian customer data while blocking only the Canadian customer data.

4. Scalability and Maintenance: Using Lake Formation's built-in capabilities to manage data access reduces the need for custom scripts or additional IAM role configurations, which can become complex and harder to maintain as the data lake grows. This approach scales well with the data size and user base.

In contrast, the other options involve more complexity or don't directly address the problem as efficiently:

- Option B suggests creating an IAM role, which would require managing and applying specific permissions and could become cumbersome as data or user requirements change. - Option C suggests column-level filtering, which is not suitable because the issue is with specific rows (i.e., those with Canadian addresses), not columns. - Option D involves tagging, which would require manually tagging each row and setting up a system to interpret these tags, adding complexity and potential for error.

Thus, setting a row-level filter is the most efficient and straightforward solution to ensure compliance with the new regulations.

Question # 124

Date: August 09, 2024

A company has implemented a lake house architecture in Amazon Redshift. The company needs to give users the ability to authenticate into Redshift query editor by using a third-party identity provider (IdP).

A data engineer must set up the authentication mechanism.

What is the first step the data engineer should take to meet this requirement?

A. Register the third-party IdP as an identity provider in the configuration settings of the Redshift cluster.
B. Register the third-party IdP as an identity provider from within Amazon Redshift.
C. Register the third-party IdP as an identity provider for AVS Secrets Manager. Configure Amazon Redshift to use Secrets Manager to manage user credentials.
D. Register the third-party IdP as an identity provider for AWS Certificate Manager (ACM). Configure Amazon Redshift to use ACM to manage user credentials.

Correct Answer:

A. Register the third-party IdP as an identity provider in the configuration settings of the Redshift cluster.

B. Register the third-party IdP as an identity provider from within Amazon Redshift.

C. Register the third-party IdP as an identity provider for AVS Secrets Manager. Configure Amazon Redshift to use Secrets Manager to manage user credentials.

D. Register the third-party IdP as an identity provider for AWS Certificate Manager (ACM). Configure Amazon Redshift to use ACM to manage user credentials.

Explanation:

The task at hand is to allow users to authenticate into Amazon Redshift's query editor using a third-party identity provider (IdP). To achieve this, the first step is to register the third-party IdP with Amazon Redshift. Let's break down why option A is the correct choice:

1. Understanding the Requirement: The company wants users to authenticate into Amazon Redshift using a third-party IdP. This involves setting up a Single Sign-On (SSO) mechanism, where the third-party IdP handles the authentication and provides access tokens to Amazon Redshift.

2. Role of Third-Party IdP: An IdP is a service that creates, maintains, and manages identity information for users and provides authentication services to applications. By integrating a third-party IdP, the company allows users to log in using their existing credentials managed by the IdP, enhancing security and user convenience.

3. Amazon Redshift Configuration: To integrate a third-party IdP, you must first register it with the Redshift cluster. This registration involves configuring the cluster to recognize and trust the IdP for authentication purposes. Essentially, you are setting up the necessary trust relationship between Redshift and the IdP.

4. Option A Explanation: Option A suggests registering the third-party IdP as an identity provider in the configuration settings of the Redshift cluster. This is the correct initial step because it establishes the required configuration for Redshift to accept and validate authentication tokens issued by the IdP. This setup is essential to enable Redshift to authenticate users against the third-party IdP.

5. Why Other Options Are Incorrect: - Option B: It implies registering the IdP from within Amazon Redshift, which is not a separate step or interface apart from the cluster settings where you would typically configure such integrations. - Option C: Involves AWS Secrets Manager, which is used for securely storing and managing secrets such as database credentials, but it is not primarily intended for integrating third-party IdP for authentication purposes. - Option D: Mentions AWS Certificate Manager (ACM), which is used for managing SSL/TLS certificates. ACM is not relevant to the process of setting up authentication with a third-party IdP for Redshift.

In summary, the correct first step is to register the third-party IdP within the configuration settings of the Redshift cluster, as specified in option A. This step ensures that Redshift can communicate with the IdP to authenticate users, enabling the intended SSO functionality.

Question # 125

Date: August 06, 2024

A company currently uses a provisioned Amazon EMR cluster that includes general purpose Amazon EC2 instances. The EMR cluster uses EMR managed scaling between one to five task nodes for the company's long-running Apache Spark extract, transform, and load (ETL) job. The company runs the ETL job every day.

When the company runs the ETL job, the EMR cluster quickly scales up to five nodes. The EMR cluster often reaches maximum CPU usage, but the memory usage remains under 30%.

The company wants to modify the EMR cluster configuration to reduce the EMR costs to run the daily ETL job.

Which solution will meet these requirements MOST cost-effectively?

A. Increase the maximum number of task nodes for EMR managed scaling to 10.
B. Change the task node type from general purpose EC2 instances to memory optimized EC2 instances.
C. Switch the task node type from general purpose Re instances to compute optimized EC2 instances.
D. Reduce the scaling cooldown period for the provisioned EMR cluster.

Correct Answer:

A. Increase the maximum number of task nodes for EMR managed scaling to 10.

B. Change the task node type from general purpose EC2 instances to memory optimized EC2 instances.

C. Switch the task node type from general purpose Re instances to compute optimized EC2 instances.

D. Reduce the scaling cooldown period for the provisioned EMR cluster.

Explanation:

The correct answer to this question is C: Switch the task node type from general purpose EC2 instances to compute optimized EC2 instances. Let's break down why this is the most cost-effective solution:

1. Current Situation: - The company's EMR cluster is using general-purpose EC2 instances for its Apache Spark ETL job. - The cluster scales up to the maximum of 5 task nodes quickly, indicating that the workload is demanding. - CPU usage often hits maximum capacity, while memory usage remains low, under 30%.

2. Understanding the Resource Bottleneck: - The fact that CPU usage is maxing out while memory usage is low suggests that the ETL job is CPU-intensive rather than memory-intensive. - General-purpose instances provide a balanced mix of CPU, memory, and network resources, but they are not optimized for workloads that heavily favor one type of resource over the others.

3. Evaluating Options: - Option A: Increasing the maximum number of task nodes to 10 would allow for more parallel processing and could alleviate CPU saturation. However, this would also increase costs since more instances would be running, potentially doubling the number of instances. - Option B: Switching to memory-optimized instances would not be beneficial, as the current memory usage is low. This would likely lead to higher costs without addressing the CPU bottleneck. - Option D: Reducing the scaling cooldown period might allow the cluster to respond more quickly to changes in demand, but it doesn't address the underlying issue of CPU saturation and could lead to frequent scaling actions, potentially increasing costs due to overhead.

4. Why Option C is the Best: - Compute-optimized instances are designed to handle applications that require high computational power, which aligns with the company's CPU-intensive workload. - By switching to compute-optimized instances, the company can more efficiently utilize the available resources, likely needing fewer instances to achieve the same or better performance. - This change can help reduce costs because compute-optimized instances, while potentially more expensive on a per-instance basis than general-purpose instances, would handle the workload more effectively, potentially reducing the total number of instances required.

In summary, by selecting compute-optimized instances, the company can better match its instance type to the workload requirements, which will maximize CPU utilization while potentially minimizing costs. This solution directly addresses the CPU bottleneck without unnecessarily increasing memory capacity, making it the most cost-effective choice.

Question # 126

Date: September 05, 2024

A technology company currently uses Amazon Kinesis Data Streams to collect log data in real time. The company wants to use Amazon Redshift for downstream real-time queries and to enrich the log data.

Which solution will ingest data into Amazon Redshift with the LEAST operational overhead?

A. Set up an Amazon Kinesis Data Firehose delivery stream to send data to a Redshift provisioned cluster table.
B. Set up an Amazon Kinesis Data Firehose delivery stream to send data to Amazon S3. Configure a Redshift provisioned cluster to load data every minute.
C. Configure Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to send data directly to a Redshift provisioned cluster table.
D. Use Amazon Redshift streaming ingestion from Kinesis Data Streams and to present data as a materialized view.

Correct Answer:

A. Set up an Amazon Kinesis Data Firehose delivery stream to send data to a Redshift provisioned cluster table.

B. Set up an Amazon Kinesis Data Firehose delivery stream to send data to Amazon S3. Configure a Redshift provisioned cluster to load data every minute.

C. Configure Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to send data directly to a Redshift provisioned cluster table.

D. Use Amazon Redshift streaming ingestion from Kinesis Data Streams and to present data as a materialized view.

Explanation:

The correct answer is D, which involves using Amazon Redshift's streaming ingestion from Kinesis Data Streams to present data as a materialized view. Let's break down why this is the best choice with the least operational overhead:

1. Direct Integration: Amazon Redshift now supports streaming ingestion from Kinesis Data Streams, allowing Redshift to directly consume streaming data. This integration eliminates the need for intermediary services or additional data transformation steps, simplifying the data pipeline.

2. Materialized Views: By using materialized views in Redshift, you can efficiently query live data as it is ingested. Materialized views automatically refresh with the latest data, providing near real-time analytics without manual intervention. This feature reduces the complexity and overhead associated with managing data refresh rates and ensures that the data is always up-to-date.

3. Operational Overhead: This solution minimizes operational overhead because it leverages built-in capabilities of Redshift and Kinesis Data Streams. There's no need to manage additional infrastructure or complex data workflows, which is often required in other solutions that involve multiple AWS services or custom ETL processes.

4. Simplicity and Efficiency: The approach is straightforward and efficient. It doesn't require setting up and managing additional components like Amazon Kinesis Data Firehose or Amazon S3, as seen in options A and B. Nor does it involve configuring and maintaining analytics applications, as mentioned in option C.

5. Cost-Effectiveness: By minimizing the number of AWS services involved and reducing the need for data storage (as opposed to staging data in S3), this solution can be more cost-effective. You are only using the necessary compute resources in Redshift to handle the queries and the streaming ingestion.

Overall, option D offers a streamlined and efficient way to ingest and query streaming data in real time with minimal operational overhead, making it the optimal choice for the company's needs.

Question # 127

Date: August 06, 2024

A company ingests data from multiple data sources and stores the data in an Amazon S3 bucket. An AWS Glue extract, transform, and load (ETL) job transforms the data and writes the transformed data to an Amazon S3 based data lake. The company uses Amazon Athena to query the data that is in the data lake.

The company needs to identify matching records even when the records do not have a common unique identifier.

Which solution will meet this requirement?

A. Use Amazon Macie pattern matching as part of the ETL job.
B. Train and use the AWS Glue PySpark Filter class in the ETL job.
C. Partition tables and use the ETL job to partition the data on a unique identifier.
D. Train and use the AWS Lake Formation FindMatches transform in the ETL job.

Correct Answer:

A. Use Amazon Macie pattern matching as part of the ETL job.

B. Train and use the AWS Glue PySpark Filter class in the ETL job.

C. Partition tables and use the ETL job to partition the data on a unique identifier.

D. Train and use the AWS Lake Formation FindMatches transform in the ETL job.

Explanation:

The correct answer is D: Train and use the AWS Lake Formation FindMatches transform in the ETL job.

Here's why this solution is appropriate:

1. Understanding the Problem: The company needs to identify matching records even when these records lack a common unique identifier. This means traditional methods of matching data based on unique keys (like IDs) are not feasible. Instead, the solution must be able to intelligently identify similarities between records that might correspond to the same entity or event.

2. AWS Lake Formation FindMatches: This is a machine learning-based data matching feature within AWS Lake Formation. FindMatches is specifically designed to tackle the problem of identifying duplicate records or matching records across datasets that do not share a common identifier. It uses advanced algorithms to detect similarities and can learn from examples of matched and unmatched records to improve accuracy over time.

3. Why FindMatches is Suitable: - Machine Learning Capabilities: FindMatches leverages machine learning to learn from labeled examples of what constitutes a match and can apply this learning to new, unseen data. This is ideal for situations where records are similar but not identical or when they don't have explicit identifiers. - Flexibility: Unlike rule-based matching systems that require strict adherence to predefined rules, FindMatches can adapt to nuances in the data, which is crucial when records don't have a straightforward way to be matched. - Integration with AWS Glue: FindMatches can be seamlessly integrated into AWS Glue ETL jobs, making it a natural fit for the company's existing data processing pipeline. It allows the company to incorporate advanced matching techniques into their ETL process without requiring additional complex infrastructure.

4. Why Other Options are Less Suitable: - Option A (Amazon Macie): Macie is primarily used for data privacy and security, focusing on identifying sensitive data within S3 buckets. It is not designed for matching records or entity resolution. - Option B (AWS Glue PySpark Filter class): The Filter class is a basic data transformation tool that allows for filtering records based on predefined criteria. It doesn't have the capability to intelligently match records without a unique identifier. - Option C (Partition tables): Partitioning is a data organization strategy that optimizes querying and storage efficiency. It relies on predefined keys for partitioning, which wouldn't help in matching records without a common identifier.

In summary, AWS Lake Formation FindMatches is the most appropriate tool for this scenario because it is specifically designed to address the challenge of matching data without a common unique identifier using machine learning techniques. This makes it well-suited to enhance the company's ETL process with the necessary matching capabilities.

Question # 128

Date: September 19, 2024

A company maintains a data warehouse in an on-premises Oracle database. The company wants to build a data lake on AWS. The company wants to load data warehouse tables into Amazon S3 and synchronize the tables with incremental data that arrives from the data warehouse every day.

Each table has a column that contains monotonically increasing values. The size of each table is less than 50 GB. The data warehouse tables are refreshed every night between 1 AM and 2 AM. A business intelligence team queries the tables between 10 AM and 8 PM every day.

Which solution will meet these requirements in the MOST operationally efficient way?

A. Use an AWS Database Migration Service (AWS DMS) full load plus CDC job to load tables that contain monotonically increasing data columns from the on-premises data warehouse to Amazon S3. Use custom logic in AWS Glue to append the daily incremental data to a full-load copy that is in Amazon S3.
B. Use an AWS Glue Java Database Connectivity (JDBC) connection. Configure a job bookmark for a column that contains monotonically increasing values. Write custom logic to append the daily incremental data to a full-load copy that is in Amazon S3.

C. Use an AWS Database Migration Service (AWS DMS) full load migration to load the data warehouse tables into Amazon S3 every day. Overwrite the previous day's full-load copy every day.
D. Use AWS Glue to load a full copy of the data warehouse tables into Amazon S3 every day. Overwrite the previous day's full-load copy every day.

Correct Answer:

A. Use an AWS Database Migration Service (AWS DMS) full load plus CDC job to load tables that contain monotonically increasing data columns from the on-premises data warehouse to Amazon S3. Use custom logic in AWS Glue to append the daily incremental data to a full-load copy that is in Amazon S3.

B. Use an AWS Glue Java Database Connectivity (JDBC) connection. Configure a job bookmark for a column that contains monotonically increasing values. Write custom logic to append the daily incremental data to a full-load copy that is in Amazon S3.

C. Use an AWS Database Migration Service (AWS DMS) full load migration to load the data warehouse tables into Amazon S3 every day. Overwrite the previous day's full-load copy every day.

D. Use AWS Glue to load a full copy of the data warehouse tables into Amazon S3 every day. Overwrite the previous day's full-load copy every day.

Explanation:

The correct answer is A: Use an AWS Database Migration Service (AWS DMS) full load plus CDC job to load tables that contain monotonically increasing data columns from the on-premises data warehouse to Amazon S3. Use custom logic in AWS Glue to append the daily incremental data to a full-load copy that is in Amazon S3.

Let's break down why this is the best solution:

1. AWS Database Migration Service (DMS) with Full Load Plus CDC: - AWS DMS is specifically designed to help migrate databases to AWS with minimal downtime. It supports continuous data replication which is ideal for this scenario. - The "full load plus CDC" (Change Data Capture) approach allows you to initially load the entire dataset from the on-premises Oracle database to Amazon S3 and then continuously capture changes. This is efficient because it ensures that only the incremental changes are sent after the initial full load, reducing the data transfer load and time.

2. Handling Monotonically Increasing Columns: - The problem states that each table has a column with monotonically increasing values. This is useful for change data capture because DMS can easily track changes by identifying new rows based on this increasing column.

3. Operational Efficiency: - By using DMS for both the initial full load and the subsequent incremental updates (via CDC), you reduce the need for complex logic to identify changes manually. This is operationally efficient as it automates the synchronization process and reduces the risk of human error. - Automating the data capture and load process with AWS DMS minimizes manual intervention and ensures that the data in Amazon S3 is kept up-to-date with the on-premises data warehouse.

4. AWS Glue for Custom Logic: - AWS Glue can be used to transform and move the data within AWS. After DMS has populated Amazon S3 with both the initial full load and incremental changes, AWS Glue can be used to apply any necessary transformations or business logic to prepare the data for analytics. - AWS Glue's serverless nature means you only pay for the computing resources you use when the jobs run, making it cost-effective.

5. Alternative Options: - Option B suggests using AWS Glue with a JDBC connection for incremental data, but this requires custom logic and lacks the built-in CDC capabilities of DMS. - Options C and D propose overwriting the full data set daily, which is inefficient. This approach incurs unnecessary data transfer costs and increases the time needed to refresh the data, making it less suitable for incremental updates and operational efficiency.

By combining AWS DMS with AWS Glue, Option A provides a streamlined, automated solution for loading and synchronizing data, making it the most operationally efficient choice for the company's requirements.

Question # 129

Date: August 14, 2024

A data engineer set up an AWS Lambda function to read an object that is stored in an Amazon S3 bucket. The object is encrypted by an AWS KMS key.

The data engineer configured the Lambda function's execution role to access the S3 bucket. However, the Lambda function encountered an error and failed to retrieve the content of the object.

What is the likely cause of the error?

A. The data engineer misconfigured the permissions of the S3 bucket. The Lambda function could not access the object.
B. The Lambda function is using an outdated SDK version, which caused the read failure.
C. The S3 bucket is located in a different AWS Region than the Region where the data engineer works. Latency issues caused the Lambda function to encounter an error.
D. The Lambda function's execution role does not have the necessary permissions to access the KMS key that can decrypt the S3 object.

Correct Answer:

A. The data engineer misconfigured the permissions of the S3 bucket. The Lambda function could not access the object.

B. The Lambda function is using an outdated SDK version, which caused the read failure.

C. The S3 bucket is located in a different AWS Region than the Region where the data engineer works. Latency issues caused the Lambda function to encounter an error.

D. The Lambda function's execution role does not have the necessary permissions to access the KMS key that can decrypt the S3 object.

Explanation:

The correct answer is D: The Lambda function's execution role does not have the necessary permissions to access the KMS key that can decrypt the S3 object.

Here's why this is the right answer:

When an object in an Amazon S3 bucket is encrypted with a KMS key, accessing that object requires specific permissions to both read the object from the S3 bucket and decrypt it using the KMS key. In AWS, permissions are managed through IAM roles and policies.

1. S3 Access: The Lambda function's execution role must have permissions to access the S3 bucket. This is usually done by granting the role permissions such as `s3:GetObject`. According to the scenario, the data engineer configured the Lambda function's execution role to access the S3 bucket. This implies that the S3 access permissions are likely correctly set.

2. KMS Permissions: In addition to S3 permissions, the execution role also needs specific permissions to use the KMS key to decrypt the object. This usually involves permissions like `kms:Decrypt`. If these permissions are not granted, the Lambda function will be unable to decrypt the object, even if it can access it in the S3 bucket.

Given that the object is encrypted with a KMS key, the most likely cause of the error is that the necessary KMS permissions were not included in the Lambda function's execution role. Without the `kms:Decrypt` permission, the Lambda function can access the encrypted file but cannot decrypt it to read its contents.

The other options are less likely causes of the error for the following reasons:

- Option A: If there were issues with the S3 bucket permissions, the Lambda function wouldn't even be able to access the object to attempt decryption.

- Option B: An outdated SDK version could potentially cause issues, but it is unlikely to be specifically related to decrypting a KMS-encrypted object if the error message pointed to decryption issues.

- Option C: Latency issues due to AWS Regions would typically not cause an error specifically related to decryption failures. AWS services are designed to handle cross-region operations, although they may be slower.

Therefore, the correct and most likely reason for the Lambda function's failure is missing KMS permissions, making D the right answer.

Question # 130

Date: January 20, 2024

A data engineer is building a data pipeline on AWS by using AWS Glue extract, transform, and load (ETL) jobs. The data engineer needs to process data from Amazon RDS and MongoDB, perform transformations, and load the transformed data into Amazon Redshift for analytics. The data updates must occur every hour.

Which combination of tasks will meet these requirements with the LEAST operational overhead? (Choose two.)

A. Configure AWS Glue triggers to run the ETL jobs every hour.
B. Use AWS Glue DataBrew to clean and prepare the data for analytics.
C. Use AWS Lambda functions to schedule and run the ETL jobs every hour.
D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.
E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Correct Answer:

A. Configure AWS Glue triggers to run the ETL jobs every hour.

B. Use AWS Glue DataBrew to clean and prepare the data for analytics.

C. Use AWS Lambda functions to schedule and run the ETL jobs every hour.

D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.

E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Explanation:

Certainly! Let's break down why options A and D are the best choices for achieving the data pipeline requirements with the least operational overhead.

Option A: Configure AWS Glue triggers to run the ETL jobs every hour.

AWS Glue is a fully managed ETL (extract, transform, load) service that automates much of the operational overhead involved in managing data workflows. By using AWS Glue triggers, you can automate the scheduling of ETL jobs. Triggers in AWS Glue allow you to set up a schedule for your jobs to run at specified intervals, in this case, every hour. This approach is straightforward and minimizes manual intervention, as you don't need to set up additional scheduling mechanisms or manage the execution of scripts. The use of AWS Glue triggers is a native feature that integrates seamlessly with the Glue environment, offering a simple and efficient way to automate job execution.

Option D: Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.

AWS Glue connections are used to define how AWS Glue jobs should connect to various data sources, such as Amazon RDS and MongoDB, as well as data targets like Amazon Redshift. By using Glue connections, you reduce the need to write custom code to manage these connections, which can be complex and error-prone. Glue connections handle authentication and network settings for you, thus significantly reducing the operational overhead. Moreover, AWS Glue natively supports a wide range of data sources, which simplifies the process of moving data across different platforms. This capability aligns perfectly with the requirement to connect multiple data sources and destinations efficiently.

In contrast, the other options either introduce unnecessary complexity or are not ideally suited for the task:

- Option B suggests using AWS Glue DataBrew, which is more focused on data preparation and cleaning tasks rather than regular ETL processes, making it less suitable for this specific use case. - Option C involves using AWS Lambda functions to schedule ETL jobs, which is not the most efficient method when AWS Glue triggers can handle this natively. Using Lambda for scheduling adds extra components to manage and monitor. - Option E involves using the Redshift Data API for loading data, which is unnecessary if AWS Glue can directly load data into Redshift through its native integrations.

By choosing options A and D, you leverage AWS Glue's built-in capabilities to automate and simplify the ETL process, minimizing the need for additional components and reducing operational overhead.

Question # 131

Date: August 14, 2024

Two developers are working on separate application releases. The developers have created feature branches named Branch A and Branch B by using a GitHub repository’s master branch as the source.

The developer for Branch A deployed code to the production system. The code for Branch B will merge into a master branch in the following week’s scheduled application release.

Which command should the developer for Branch B run before the developer raises a pull request to the master branch?

A. git diff branchB mastergit commit -m
B. git pull master
C. git rebase master
D. git fetch -b master

Correct Answer:

A. git diff branchB mastergit commit -m

B. git pull master

C. git rebase master

D. git fetch -b master

Explanation:

In this scenario, the developers are working with Git, a version control system, to manage changes in their application code. Let's break down the situation:

1. Branches and Merging: The developers have created two feature branches, Branch A and Branch B, from the master branch. Branch A has already been deployed to production, meaning its changes are now part of the master branch. Branch B is preparing for its release, which involves merging its changes into the master branch.

2. Objective for Branch B: Before merging Branch B into the master branch, the developer needs to ensure that Branch B is compatible with the latest changes in the master branch. This is crucial because Branch A's changes are already merged into the master branch, and Branch B needs to be aligned with these updates to avoid conflicts and ensure smooth integration.

3. Git Rebase Command: The command `git rebase master` (Option C) is the correct choice for this situation. Here's why:

- Rebasing: Rebasing is a process in Git that allows you to move or combine a sequence of commits to a new base commit. When you run `git rebase master` while on Branch B, Git will take the changes from Branch B and replay them on top of the latest commit in the master branch. This effectively integrates the latest changes from the master branch into Branch B without creating a new merge commit. - Benefits of Rebasing: By rebasing, the developer for Branch B can ensure that any changes from Branch A (now in master) are incorporated into Branch B, resolving any potential conflicts locally before raising a pull request. This makes the commit history cleaner and avoids the "merge commit clutter" that can happen with frequent merges.

4. Other Options: Let's briefly look at why the other options are not suitable:

- Option A (`git diff branchB mastergit commit -m`): This command seems incorrect or malformed. `git diff` shows the differences between branches but doesn't integrate changes. Additionally, `git commit -m` without relevant modifications does not accomplish re-synchronization with the master branch.

- Option B (`git pull master`): This command is incomplete. `git pull` is used to fetch changes from a remote repository and merge them into the current branch. It requires specifying the remote or using the correct syntax like `git pull origin master`. Also, it creates a merge commit, which might complicate the history.

- Option D (`git fetch -b master`): This is not a valid Git command. `git fetch` is used to update local references to remote branches but does not merge changes. The `-b` option is incorrectly used here.

In summary, `git rebase master` is the appropriate command for the developer of Branch B to ensure their branch is up-to-date and conflict-free with the latest changes from the master branch before raising a pull request. This process helps maintain a clean and linear commit history, which is often preferred in many development workflows.

Question # 132

Date: August 06, 2024

A company stores employee data in Amazon Resdshift. A table names Employee uses columns named Region ID, Department ID, and Role ID as a compound sort key.

Which queries will MOST increase the speed of query by using a compound sort key of the table? (Choose two.)

A. Select *from Employee where Region ID='North America';
B. Select *from Employee where Region ID='North America' and Department ID=20;
C. Select *from Employee where Department ID=20 and Region ID='North America';
D. Select *from Employee where Role ID=50;
E. Select *from Employee where Region ID='North America' and Role ID=50;

Correct Answer:

A. Select *from Employee where Region ID='North America';

B. Select *from Employee where Region ID='North America' and Department ID=20;

C. Select *from Employee where Department ID=20 and Region ID='North America';

D. Select *from Employee where Role ID=50;

E. Select *from Employee where Region ID='North America' and Role ID=50;

Explanation:

Sure! Let's dive into why options B and E are the best choices for optimizing query speed using a compound sort key in Amazon Redshift.

Understanding Compound Sort Keys: In Amazon Redshift, a compound sort key is a method to define the order in which the data is physically stored on disk. When a compound sort key is used, the data is sorted by the first column in the key, then by the second column, and so on. This order can significantly affect the performance of queries, especially when filtering data.

Given Compound Sort Key: For the `Employee` table, the compound sort key is defined as `Region ID`, `Department ID`, and `Role ID`. This means the table data is first sorted by `Region ID`, then within each `Region ID`, it is sorted by `Department ID`, and finally, within each `Department ID`, it is sorted by `Role ID`.

Evaluating the Options:

- Option A: `Select from Employee where Region ID='North America';` - This query only uses the first column of the compound sort key (`Region ID`). While this does leverage the sort key, it does not take full advantage of the compound structure to further narrow down the data, making it less optimal compared to options that use more of the sort key columns.

- Option B: `Select from Employee where Region ID='North America' and Department ID=20;` - This query uses both the first (`Region ID`) and the second (`Department ID`) columns of the sort key. By doing so, it can efficiently narrow down the data to those sorted by `Region ID` and further filter within each `Region ID` by `Department ID`. This makes it highly efficient and increases query speed.

- Option C: `Select from Employee where Department ID=20 and Region ID='North America';` - Although this query involves both `Region ID` and `Department ID`, the order of conditions does not match the sort key order. Redshift optimizes based on the order of the sort key, so this query is less efficient than option B.

- Option D: `Select from Employee where Role ID=50;` - This query only uses the third column of the compound sort key (`Role ID`). Since it does not leverage the sorting by the first or second columns, it cannot take full advantage of the compound sort key structure for faster query performance.

- Option E: `Select from Employee where Region ID='North America' and Role ID=50;` - This query uses the first (`Region ID`) and the third (`Role ID`) columns of the sort key. Although it skips the second column (`Department ID`), it still benefits from starting with the first column of the sort key, making it more efficient than queries that do not use the first column at all.

Conclusion: Options B and E are correct because they utilize the compound sort key effectively. Option B aligns with the order of the sort key more closely, while option E still benefits from using the leading column of the sort key. Both options enhance query speed by reducing the amount of data that needs to be scanned, as they leverage the physical data ordering provided by the sort key.

Question # 133

Date: August 14, 2024

A company receives test results from testing facilities that are located around the world. The company stores the test results in millions of 1 KB JSON files in an Amazon S3 bucket. A data engineer needs to process the files, convert them into Apache Parquet format, and load them into Amazon Redshift tables. The data engineer uses AWS Glue to process the files, AWS Step Functions to orchestrate the processes, and Amazon EventBridge to schedule jobs.

The company recently added more testing facilities. The time required to process files is increasing. The data engineer must reduce the data processing time.

Which solution will MOST reduce the data processing time?

A. Use AWS Lambda to group the raw input files into larger files. Write the larger files back to Amazon S3. Use AWS Glue to process the files. Load the files into the Amazon Redshift tables.
B. Use the AWS Glue dynamic frame file-grouping option to ingest the raw input files. Process the files. Load the files into the Amazon Redshift tables.
C. Use the Amazon Redshift COPY command to move the raw input files from Amazon S3 directly into the Amazon Redshift tables. Process the files in Amazon Redshift.
D. Use Amazon EMR instead of AWS Glue to group the raw input files. Process the files in Amazon EMR. Load the files into the Amazon Redshift tables.

Correct Answer:

A. Use AWS Lambda to group the raw input files into larger files. Write the larger files back to Amazon S3. Use AWS Glue to process the files. Load the files into the Amazon Redshift tables.

B. Use the AWS Glue dynamic frame file-grouping option to ingest the raw input files. Process the files. Load the files into the Amazon Redshift tables.

C. Use the Amazon Redshift COPY command to move the raw input files from Amazon S3 directly into the Amazon Redshift tables. Process the files in Amazon Redshift.

D. Use Amazon EMR instead of AWS Glue to group the raw input files. Process the files in Amazon EMR. Load the files into the Amazon Redshift tables.

Explanation:

The correct answer is B: Use the AWS Glue dynamic frame file-grouping option to ingest the raw input files. Process the files. Load the files into the Amazon Redshift tables.

Here's why this is the most effective solution:

1. AWS Glue Dynamic Frame File-Grouping: AWS Glue provides a feature called dynamic frames that can automatically group smaller files into larger partitions or files. This feature optimizes the read and write operations by reducing the overhead associated with processing numerous small files. When dealing with millions of 1 KB JSON files, the overhead of handling each file individually can significantly increase processing time. By grouping these files, AWS Glue reduces the number of read/write operations, thus speeding up the processing time.

2. Seamless Integration: AWS Glue is designed to work seamlessly with other AWS services like Amazon S3 and Amazon Redshift. By leveraging the dynamic frame file-grouping feature, you can continue using AWS Glue for both the transformation and loading stages without introducing additional complexity or services that might require extra management or coding effort.

3. Optimized for Big Data: AWS Glue is specifically designed for ETL (Extract, Transform, Load) operations in big data environments. It can handle large-scale data processing tasks efficiently, and the dynamic frame feature further enhances its ability to deal with scenarios involving numerous small files.

4. Maintainability and Simplicity: This solution keeps the architecture simple by utilizing built-in features of AWS Glue rather than introducing additional components or services. This simplicity helps maintain the system and reduces the possibility of errors or complications that might arise from integrating and managing more services.

5. Cost-Effectiveness: While AWS Glue may incur costs for processing data, the efficiency gained by using dynamic frames to reduce the processing time can translate into cost savings. This is because the quicker the processing, the less time you will spend on compute resources, which often is a major cost factor in such operations.

In contrast:

- Option A suggests using AWS Lambda to group files, which might not be the most efficient or cost-effective choice for handling large numbers of small files due to Lambda's execution time limitations and the potential complexity of managing intermediate file states. - Option C proposes loading raw JSON files directly into Amazon Redshift, which might not be efficient for processing and converting them into Parquet format. Redshift is not optimized for processing raw data formats like JSON directly at scale.

- Option D introduces Amazon EMR for file grouping, which adds complexity by involving another service. While EMR can handle big data, using AWS Glue's built-in capabilities is simpler and more directly aligned with the existing workflow.

Therefore, B is the best choice as it effectively reduces processing time while maintaining simplicity and integration with the existing AWS infrastructure.

Question # 134

Date: August 06, 2024

A data engineer uses Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to run data pipelines in an AWS account.

A workflow recently failed to run. The data engineer needs to use Apache Airflow logs to diagnose the failure of the workflow.

Which log type should the data engineer use to diagnose the cause of the failure?

A. YourEnvironmentName-WebServer
B. YourEnvironmentName-Scheduler
C. YourEnvironmentName-DAGProcessing
D. YourEnvironmentName-Task

Correct Answer:

A. YourEnvironmentName-WebServer

B. YourEnvironmentName-Scheduler

C. YourEnvironmentName-DAGProcessing

D. YourEnvironmentName-Task

Explanation:

To diagnose a failed workflow in Amazon Managed Workflows for Apache Airflow (Amazon MWAA), it's crucial to check the logs that provide the most relevant information about the tasks within the workflow. Here's why option D, "YourEnvironmentName-Task," is the correct log type to use:

1. Understanding Apache Airflow Workflows: In Apache Airflow, a workflow is defined as a Directed Acyclic Graph (DAG) consisting of multiple tasks. Each task represents a unit of work or a step in the workflow. When diagnosing a failure, it's essential to identify which specific task failed and why.

2. Task Logs: The "YourEnvironmentName-Task" log type contains logs specific to the tasks executed in the workflow. These logs provide detailed information about the execution of each task, including any errors or exceptions that occurred during task execution. This makes them the most direct source of information for understanding and diagnosing task failures within a DAG.

3. Other Log Types: - WebServer Logs (A): These logs relate to the web server component of Airflow, which handles the user interface and API requests. They are not typically useful for diagnosing task failures. - Scheduler Logs (B): The scheduler logs are concerned with the scheduling of tasks but do not provide detailed information about task execution failures. They are more about the orchestration and timing of task runs. - DAG Processing Logs (C): These logs detail the processing of DAGs themselves, such as parsing and loading, but they do not cover the execution of individual tasks within those DAGs.

4. Specificity of Task Logs: Since task logs are specifically designed to capture what happens during the execution of each task, they include stack traces, error messages, and any other relevant output that can pinpoint what went wrong. This level of detail is what a data engineer needs to effectively troubleshoot and resolve issues with a failed workflow.

In summary, to diagnose the cause of a workflow failure in Amazon MWAA, the task logs ("YourEnvironmentName-Task") are the most pertinent as they directly capture and provide insights into the execution and any issues encountered by the individual tasks within the workflow.

Question # 135

Date: September 19, 2024

A finance company uses Amazon Redshift as a data warehouse. The company stores the data in a shared Amazon S3 bucket. The company uses Amazon Redshift Spectrum to access the data that is stored in the S3 bucket. The data comes from certified third-party data providers. Each third-party data provider has unique connection details.

To comply with regulations, the company must ensure that none of the data is accessible from outside the company's AWS environment.

Which combination of steps should the company take to meet these requirements? (Choose two.)

A. Replace the existing Redshift cluster with a new Redshift cluster that is in a private subnet. Use an interface VPC endpoint to connect to the Redshift cluster. Use a NAT gateway to give Redshift access to the S3 bucket.
B. Create an AWS CloudHSM hardware security module (HSM) for each data provider. Encrypt each data provider's data by using the corresponding HSM for each data provider.
C. Turn on enhanced VPC routing for the Amazon Redshift cluster. Set up an AWS Direct Connect connection and configure a connection between each data provider and the finance company's VP
D. Define table constraints for the primary keys and the foreign keys.
E. Use federated queries to access the data from each data provider. Do not upload the data to the S3 bucket. Perform the federated queries through a gateway VPC endpoint.

Correct Answer:

A. Replace the existing Redshift cluster with a new Redshift cluster that is in a private subnet. Use an interface VPC endpoint to connect to the Redshift cluster. Use a NAT gateway to give Redshift access to the S3 bucket.

B. Create an AWS CloudHSM hardware security module (HSM) for each data provider. Encrypt each data provider's data by using the corresponding HSM for each data provider.

C. Turn on enhanced VPC routing for the Amazon Redshift cluster. Set up an AWS Direct Connect connection and configure a connection between each data provider and the finance company’s VP

D. Define table constraints for the primary keys and the foreign keys.

E. Use federated queries to access the data from each data provider. Do not upload the data to the S3 bucket. Perform the federated queries through a gateway VPC endpoint.

Explanation:

The correct answer to this problem is "A" and "C". Here's why these options are appropriate steps to ensure that the data is secure and only accessible within the company's AWS environment:

A. Replace the existing Redshift cluster with a new Redshift cluster that is in a private subnet. Use an interface VPC endpoint to connect to the Redshift cluster. Use a NAT gateway to give Redshift access to the S3 bucket.

This option involves setting up the Redshift cluster within a private subnet. By placing it in a private subnet, you ensure that the cluster is not directly accessible from the internet, adding a layer of security. An interface VPC endpoint allows you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection, which further enhances security by keeping the traffic within the AWS network. Using a NAT gateway allows resources in the private subnet to access the S3 bucket for data queries and retrievals without exposing the Redshift cluster to the internet, complying with the requirement to keep data access within the company's AWS environment.

C. Turn on enhanced VPC routing for the Amazon Redshift cluster. Set up an AWS Direct Connect connection and configure a connection between each data provider and the finance company’s VPC.

Enhanced VPC routing forces all COPY and UNLOAD traffic between your cluster and your data repositories through your VPC. When you enable this, Redshift uses your VPC for all communication with S3, which means that the data traffic remains within your VPC and does not traverse the public internet. This is crucial for ensuring compliance with regulations that mandate the data remains accessible only from within the AWS environment. AWS Direct Connect provides a dedicated network connection from the third-party data provider’s location to the finance company’s AWS environment, further ensuring that data transfers happen securely and privately, bypassing the public internet and adhering to regulatory requirements.

By implementing these two solutions, the finance company can ensure that their data remains secure and compliant with regulations, as the data access is confined to the company's AWS environment, preventing external access.

Question # 136

Date: September 05, 2024

Files from multiple data sources arrive in an Amazon S3 bucket on a regular basis. A data engineer wants to ingest new files into Amazon Redshift in near real time when the new files arrive in the S3 bucket.

Which solution will meet these requirements?

A. Use the query editor v2 to schedule a COPY command to load new files into Amazon Redshift.
B. Use the zero-ETL integration between Amazon Aurora and Amazon Redshift to load new files into Amazon Redshift.
C. Use AWS Glue job bookmarks to extract, transform, and load (ETL) load new files into Amazon Redshift.
D. Use S3 Event Notifications to invoke an AWS Lambda function that loads new files into Amazon Redshift.

Correct Answer:

A. Use the query editor v2 to schedule a COPY command to load new files into Amazon Redshift.

B. Use the zero-ETL integration between Amazon Aurora and Amazon Redshift to load new files into Amazon Redshift.

C. Use AWS Glue job bookmarks to extract, transform, and load (ETL) load new files into Amazon Redshift.

D. Use S3 Event Notifications to invoke an AWS Lambda function that loads new files into Amazon Redshift.

Explanation:

The correct answer is D: Use S3 Event Notifications to invoke an AWS Lambda function that loads new files into Amazon Redshift.

Here's why this solution is appropriate:

1. Real-time Processing Requirement: The question specifies the need to ingest new files into Amazon Redshift in "near real time" as they arrive in the S3 bucket. This requirement indicates that the solution should be able to trigger actions immediately as new data becomes available, rather than relying on scheduled or batch processes.

2. S3 Event Notifications: Amazon S3 provides a feature called event notifications, which can automatically trigger actions when certain events occur in the bucket, such as when a new file is created. This capability is perfect for detecting new files as they arrive.

3. AWS Lambda: AWS Lambda is a serverless compute service that allows you to run code in response to events without managing servers. By using S3 event notifications to invoke a Lambda function, you can automatically trigger a process whenever a new file lands in the S3 bucket. This setup ensures that your data is processed in near real time.

4. Loading Data into Amazon Redshift: The Lambda function can be programmed to use the COPY command to load new data into Amazon Redshift. The COPY command is efficient for bulk-loading data from S3 to Redshift and is a common method used for this purpose.

5. Scalability and Cost-Effectiveness: Using Lambda for this task is both scalable and cost-effective. Lambda scales automatically with the number of incoming events (new files in this case), and you only pay for the compute time you consume, which can be more economical than maintaining a continuously running service.

In contrast, the other options do not meet the requirements as effectively: - Option A involves scheduling, which might not meet the "near real time" requirement since it relies on predefined intervals. - Option B involves a zero-ETL integration specific to Amazon Aurora and does not address file loading from S3. - Option C uses AWS Glue for ETL processes, which might not be as immediate as the Lambda-triggered solution and could introduce unnecessary latency.

Overall, using S3 Event Notifications with AWS Lambda provides a direct, event-driven approach to achieve near real-time ingestion of new files into Amazon Redshift, making it the most suitable solution for the given requirements.

Question # 137

Date: September 05, 2024

A company is building a data lake for a new analytics team. The company is using Amazon S3 for storage and Amazon Athena for query analysis. All data that is in Amazon S3 is in Apache Parquet format.

The company is running a new Oracle database as a source system in the company's data center. The company has 70 tables in the Oracle database. All the tables have primary keys. Data can occasionally change in the source system. The company wants to ingest the tables every day into the data lake.

Which solution will meet this requirement with the LEAST effort?

A. Create an Apache Sqoop job in Amazon EMR to read the data from the Oracle database. Configure the Sqoop job to write the data to Amazon S3 in Parquet format.
B. Create an AWS Glue connection to the Oracle database. Create an AWS Glue bookmark job to ingest the data incrementally and to write the data to Amazon S3 in Parquet format.
C. Create an AWS Database Migration Service (AWS DMS) task for ongoing replication. Set the Oracle database as the source. Set Amazon S3 as the target. Configure the task to write the data in Parquet format.
D. Create an Oracle database in Amazon RDS. Use AWS Database Migration Service (AWS DMS) to migrate the on-premises Oracle database to Amazon RDS. Configure triggers on the tables to invoke AWS Lambda functions to write changed records to Amazon S3 in Parquet format.

Correct Answer:

A. Create an Apache Sqoop job in Amazon EMR to read the data from the Oracle database. Configure the Sqoop job to write the data to Amazon S3 in Parquet format.

B. Create an AWS Glue connection to the Oracle database. Create an AWS Glue bookmark job to ingest the data incrementally and to write the data to Amazon S3 in Parquet format.

C. Create an AWS Database Migration Service (AWS DMS) task for ongoing replication. Set the Oracle database as the source. Set Amazon S3 as the target. Configure the task to write the data in Parquet format.

D. Create an Oracle database in Amazon RDS. Use AWS Database Migration Service (AWS DMS) to migrate the on-premises Oracle database to Amazon RDS. Configure triggers on the tables to invoke AWS Lambda functions to write changed records to Amazon S3 in Parquet format.

Explanation:

The correct answer is C: Create an AWS Database Migration Service (AWS DMS) task for ongoing replication. Set the Oracle database as the source. Set Amazon S3 as the target. Configure the task to write the data in Parquet format.

Here's why this solution is the most suitable with the least effort:

1. Purpose-Built Service for Data Migration: AWS DMS is specifically designed to simplify and automate data migrations between different databases, and it supports ongoing replication. This makes it a natural fit for the requirement of daily ingestion of data from an Oracle database to Amazon S3.

2. Support for Ongoing Replication: DMS can be configured for ongoing replication, which means it can continuously capture and apply changes from the source Oracle database to the target, ensuring that the data in Amazon S3 remains up-to-date with minimal manual intervention.

3. Direct Integration with S3 and Parquet Format: AWS DMS supports writing directly to Amazon S3 and can output data in Apache Parquet format. This matches the company's existing data lake format and allows seamless integration with their analytics tools, such as Amazon Athena.

4. Minimal Operational Overhead: Compared to other options, using DMS requires less ongoing maintenance and monitoring. Once set up, it automatically handles the data migration process without the need for complex custom scripts or additional infrastructure management.

5. Efficient Handling of Changes: Since the Oracle database can occasionally change, DMS's ability to handle change data capture (CDC) efficiently ensures that only the changes are migrated, reducing data transfer costs and processing time.

In contrast, the other options involve more complexity:

- Option A (Apache Sqoop and Amazon EMR) requires managing and maintaining EMR clusters and custom Sqoop jobs, which can be more complex and require significant operational effort.

- Option B (AWS Glue) involves setting up and maintaining Glue jobs, which can become complex when dealing with incremental changes and ensuring consistent data states.

- Option D (migrating to Amazon RDS and using triggers) involves additional steps of setting up and managing an RDS instance, creating and maintaining triggers, and setting up Lambda functions, which adds significant complexity and potential points of failure.

Therefore, AWS DMS provides the most straightforward, scalable, and efficient solution for this scenario with the least operational effort.

Question # 138

Date: September 08, 2024

A transportation company wants to track vehicle movements by capturing geolocation records. The records are 10 bytes in size. The company receives up to 10.000 records every second. Data transmission delays of a few minutes are acceptable because of unreliable network conditions.

The transportation company wants to use Amazon Kinesis Data Streams to ingest the geolocation data. The company needs a reliable mechanism to send data to Kinesis Data Streams. The company needs to maximize the throughput efficiency of the Kinesis shards.

Which solution will meet these requirements in the MOST operationally efficient way?

A. Kinesis Agent
B. Kinesis Producer Library (KPL)
C. Amazon Kinesis Data Firehose
D. Kinesis SDK

Correct Answer:

A. Kinesis Agent

B. Kinesis Producer Library (KPL)

C. Amazon Kinesis Data Firehose

D. Kinesis SDK

Explanation:

Sure, let's break down why the Kinesis Producer Library (KPL) is the most operationally efficient solution for this scenario.

The transportation company needs to ingest a large volume of small geolocation records (10 bytes each) into Amazon Kinesis Data Streams, with a rate of up to 10,000 records per second. Given the requirement to maximize throughput efficiency of the Kinesis shards, the KPL is the most suitable tool for several reasons:

1. Aggregation and Batching: KPL is specifically designed to aggregate and batch multiple records into a single larger record before sending it to Kinesis Data Streams. This is crucial because it helps maximize the throughput of the shards by reducing the number of individual records sent. Instead of sending each 10-byte record separately, KPL combines them into fewer, larger payloads, which is much more efficient from a network and processing perspective.

2. Latency Tolerance: The scenario allows for data transmission delays of a few minutes, which aligns well with KPL's operation. KPL can hold records for a short period to batch them optimally, aligning with the acceptable delay range.

3. Operational Efficiency: KPL manages many operational details like retries, error handling, and efficient resource utilization under the hood. This reduces the burden on the development and operations teams, allowing them to focus on other tasks rather than managing the complexities of optimizing data ingestion.

4. Throughput Maximization: By efficiently aggregating records, KPL maximizes the throughput of each shard, reducing the number of shards needed and thus optimizing cost and performance.

In contrast, other options like:

- Kinesis Agent: Best for simpler use cases like log file monitoring, but not as effective for optimizing throughput with high-volume, small-size records. - Amazon Kinesis Data Firehose: Primarily used for data transformation and loading into other AWS services. While it handles data delivery, it's not as focused on optimizing throughput for Kinesis Data Streams specifically. - Kinesis SDK: Provides lower-level control but lacks the built-in aggregation and batching capabilities of KPL, requiring more manual management and possibly leading to less efficient shard usage.

Therefore, KPL is the best choice for meeting the company's requirements efficiently and effectively.

Question # 139

Date: September 19, 2024

An investment company needs to manage and extract insights from a volume of semi-structured data that grows continuously.

A data engineer needs to deduplicate the semi-structured data, remove records that are duplicates, and remove common misspellings of duplicates.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use the FindMatches feature of AWS Glue to remove duplicate records.
B. Use non-Windows functions in Amazon Athena to remove duplicate records.
C. Use Amazon Neptune ML and an Apache Gremlin script to remove duplicate records.
D. Use the global tables feature of Amazon DynamoDB to prevent duplicate data.

Correct Answer:

A. Use the FindMatches feature of AWS Glue to remove duplicate records.

B. Use non-Windows functions in Amazon Athena to remove duplicate records.

C. Use Amazon Neptune ML and an Apache Gremlin script to remove duplicate records.

D. Use the global tables feature of Amazon DynamoDB to prevent duplicate data.

Explanation:

The correct answer is A: Use the FindMatches feature of AWS Glue to remove duplicate records.

AWS Glue is a fully managed data integration service that makes it easier to discover, prepare, and combine data for analytics, machine learning, and application development. One of its powerful features is FindMatches, which is specifically designed to identify and deduplicate records in your datasets.

Here's why this option is the best fit for the scenario:

1. Designed for Deduplication: The FindMatches feature in AWS Glue is specifically built to handle the task of identifying and removing duplicate records in datasets. It uses machine learning to detect duplicate entries, even when they are not exact matches, which is ideal for handling semi-structured data with potential misspellings or variations in data entries.

2. Low Operational Overhead: AWS Glue is a serverless service, which means you don't need to manage any infrastructure. This significantly reduces operational overhead compared to managing your own data processing pipelines. You simply configure your job and let AWS handle the scaling and execution.

3. Handling Variations: The FindMatches feature can identify duplicates even when there are minor variations or misspellings in the data. This is particularly useful for the investment company's requirement to remove common misspellings of duplicates.

4. Ease of Use: AWS Glue provides a user-friendly interface and can be easily integrated with other AWS services. It allows data engineers to build data processing jobs without needing extensive programming skills, which contributes to lower operational overhead.

5. Scalability: As the company's data grows continuously, AWS Glue can scale to accommodate increasing data volumes without requiring manual intervention or infrastructure changes.

Option B, using non-Windows functions in Amazon Athena, is more suited for querying and analyzing data rather than deduplication and would require custom queries that might not efficiently handle misspellings. Option C, involving Amazon Neptune ML and Apache Gremlin, is complex and intended for graph databases, which isn't necessary for simple deduplication tasks. Option D, using global tables in Amazon DynamoDB, is meant for data replication across regions and not specifically for deduplication tasks.

In summary, AWS Glue's FindMatches is tailored for deduplication tasks, minimizing operational overhead while efficiently handling the company's needs for cleaning semi-structured data.

Question # 140

Date: October 27, 2024

A company is building an inventory management system and an inventory reordering system to automatically reorder products. Both systems use Amazon Kinesis Data Streams. The inventory management system uses the Amazon Kinesis Producer Library (KPL) to publish data to a stream. The inventory reordering system uses the Amazon Kinesis Client Library (KCL) to consume data from the stream. The company configures the stream to scale up and down as needed.

Before the company deploys the systems to production, the company discovers that the inventory reordering system received duplicated data.

Which factors could have caused the reordering system to receive duplicated data? (Choose two.)

A. The producer experienced network-related timeouts.
B. The stream's value for the IteratorAgeMilliseconds metric was too high.
C. There was a change in the number of shards, record processors, or both.
D. The AggregationEnabled configuration property was set to true.
E. The max_records configuration property was set to a number that was too high.

Correct Answer:

A. The producer experienced network-related timeouts.

B. The stream's value for the IteratorAgeMilliseconds metric was too high.

C. There was a change in the number of shards, record processors, or both.

D. The AggregationEnabled configuration property was set to true.

E. The max_records configuration property was set to a number that was too high.

Explanation:

Certainly! In this scenario, the company is dealing with duplicated data in their inventory reordering system, which uses Amazon Kinesis Data Streams. Let's explore why certain factors might lead to duplicated data:

1. The producer experienced network-related timeouts (Option A): When a producer, such as the one using the Amazon Kinesis Producer Library (KPL), sends data to a Kinesis stream, network-related timeouts can result in the producer not receiving an acknowledgment that the data was successfully sent. In such cases, the KPL might attempt to resend the data, leading to potential duplication. This is because the producer cannot confirm whether the initial data was successfully received and processed by the stream, so it errs on the side of caution and resends the data.

2. The stream's value for the IteratorAgeMilliseconds metric was too high (Option B): The IteratorAgeMilliseconds metric indicates the age of the last record returned by the GetRecords call of a shard. While a high value may suggest that records are being processed slowly, it does not directly cause duplication. Instead, it indicates latency or backlog issues. Therefore, this option is not directly related to causing duplication of records.

3. There was a change in the number of shards, record processors, or both (Option C): Changes in the number of shards or record processors can lead to duplicate data processing. When shards are split or merged, or when the distribution of workload across record processors changes, the KCL might reassign shards to different consumers. During this reassignment process, some records might be reprocessed, leading to duplication. This is a known behavior when dealing with resharding events, as the state of processing might not be perfectly synchronized during these transitions.

4. The AggregationEnabled configuration property was set to true (Option D): Aggregation is a feature of KPL that allows multiple logical user records to be sent to Kinesis Data Streams as a single Kinesis record. While enabling aggregation helps in reducing costs and increasing throughput, it does not inherently cause duplication. Instead, it affects how records are packed together before being sent to the stream. Therefore, this is unlikely to be a cause of duplicate data.

5. The max_records configuration property was set to a number that was too high (Option E): This property controls the maximum number of records that can be returned by a single GetRecords call in the KCL. Setting this value too high can lead to increased latency in processing but does not directly cause duplication. The duplication would typically arise from issues in data acknowledgment and replay, not the number of records being fetched at once.

Given these explanations, the factors most likely to cause duplicated data in this scenario are:

- Option A (The producer experienced network-related timeouts): This can lead to data being sent more than once due to retry logic. - Option C (There was a change in the number of shards, record processors, or both): This can result in records being reprocessed during resharding events.

These are the factors that align with common causes of data duplication in systems using Kinesis Data Streams with KPL and KCL.

Question # 141

Date: September 19, 2024

A mobile gaming company wants to capture data from its gaming app. The company wants to make the data available to three internal consumers of the data. The data records are approximately 20 KB in size.

The company wants to achieve optimal throughput from each device that runs the gaming app. Additionally, the company wants to develop an application to process data streams. The stream-processing application must have dedicated throughput for each internal consumer.

Which solution will meet these requirements?

A. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Use the enhanced fan-out feature with a stream for each internal consumer.
B. Configure the mobile app to call the PutRecordBatch API operation to send data to Amazon Kinesis Data Firehose. Submit an AWS Support case to turn on dedicated throughput for the company's AWS account. Allow each internal consumer to access the stream.
C. Configure the mobile app to use the Amazon Kinesis Producer Library (KPL) to send data to Amazon Kinesis Data Firehose. Use the enhanced fan-out feature with a stream for each internal consumer.
D. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Host the stream-processing application for each internal consumer on Amazon EC2 instances. Configure auto scaling for the EC2 instances.

Correct Answer:

A. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Use the enhanced fan-out feature with a stream for each internal consumer.

B. Configure the mobile app to call the PutRecordBatch API operation to send data to Amazon Kinesis Data Firehose. Submit an AWS Support case to turn on dedicated throughput for the company's AWS account. Allow each internal consumer to access the stream.

C. Configure the mobile app to use the Amazon Kinesis Producer Library (KPL) to send data to Amazon Kinesis Data Firehose. Use the enhanced fan-out feature with a stream for each internal consumer.

D. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Host the stream-processing application for each internal consumer on Amazon EC2 instances. Configure auto scaling for the EC2 instances.

Explanation:

The correct answer is A, which involves configuring the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams and using the enhanced fan-out feature with a stream for each internal consumer. Let’s break down why this solution is the best choice given the requirements of the mobile gaming company:

1. Data Size and Throughput Requirements: The data records are approximately 20 KB in size, which is well within the capabilities of Kinesis Data Streams. Kinesis Data Streams is designed for handling large volumes of streaming data with low latency, making it suitable for applications like mobile gaming where real-time data capture is crucial.

2. PutRecords API Operation: Using the PutRecords API operation is ideal for sending multiple records to Kinesis Data Streams in a single HTTP request. This is more efficient than sending individual records, reducing the overhead and increasing throughput, which aligns with the company's goal of achieving optimal throughput from each device.

3. Enhanced Fan-Out: The enhanced fan-out feature of Kinesis Data Streams allows for dedicated throughput per consumer. This means each internal consumer can have their own dedicated read throughput, reducing the risk of consumers competing for bandwidth and ensuring that each can process data independently and efficiently. This is critical when multiple internal consumers need to process the data simultaneously without affecting each other’s performance.

4. Stream for Each Internal Consumer: By setting up a stream for each internal consumer, the company can ensure that each consumer's data processing needs are met without interference. This setup allows for scaling and customizing the data processing pipeline according to each consumer's specific requirements, further optimizing the throughput and processing capabilities.

5. Real-time Processing: Kinesis Data Streams is designed for real-time data processing. This aligns with the company's needs to develop an application to process data streams in real-time, enabling them to react quickly to the data being generated by the gaming app.

Option B suggests using Kinesis Data Firehose, which is more suited for loading data into AWS data stores rather than real-time processing. Option C incorrectly mentions using the enhanced fan-out feature with Firehose, which is not applicable. Option D involves using EC2 instances for stream processing, which adds unnecessary complexity and overhead compared to using the enhanced fan-out feature directly with Kinesis Data Streams.

Overall, option A meets all the requirements by providing efficient data transmission, dedicated throughput for each consumer, and the ability to handle real-time data processing, making it the most suitable solution for the company's needs.

Question # 142

Date: October 27, 2024

A data engineer needs to create a new empty table in Amazon Athena that has the same schema as an existing table named old_table.

Which SQL statement should the data engineer use to meet this requirement?

A. CREATE TABLE new_table AS SELECT * FROM old_tables;
B. INSERT INTO new_table SELECT * FROM old_table;
C. CREATE TABLE new_table (LIKE old_table);
D. CREATE TABLE new_table AS (SELECT * FROM old_table) WITH NO DATA;

Correct Answer:

A. CREATE TABLE new_table AS SELECT * FROM old_tables;

B. INSERT INTO new_table SELECT * FROM old_table;

C. CREATE TABLE new_table (LIKE old_table);

D. CREATE TABLE new_table AS (SELECT * FROM old_table) WITH NO DATA;

Explanation:

The task here is to create a new table in Amazon Athena that has the same schema as an existing table, `old_table`, but without any data from `old_table`.

Let's evaluate the options:

A. CREATE TABLE new_table AS SELECT FROM old_table; - This statement would create a new table `new_table` and populate it with all the data from `old_table`. It doesn't meet the requirement of creating an empty table, as it transfers both the schema and the data.

B. INSERT INTO new_table SELECT FROM old_table; - This statement assumes that `new_table` already exists. It is used to copy data from `old_table` to `new_table`, which again doesn't align with the requirement since we want to create a new, empty table.

C. CREATE TABLE new_table (LIKE old_table); - This SQL syntax is not valid in Amazon Athena. While some databases may support a `LIKE` clause to copy schema, Athena does not recognize this syntax, therefore making it an incorrect choice.

D. CREATE TABLE new_table AS (SELECT FROM old_table) WITH NO DATA; - This is the correct option. The `CREATE TABLE AS SELECT` (CTAS) statement in Amazon Athena allows you to create a new table by querying an existing table. The `WITH NO DATA` clause means that only the schema from `old_table` is copied, and no data is transferred to `new_table`. This perfectly aligns with the requirement to create an empty table that has the same schema as `old_table`.

In summary, option D correctly uses Athena's CTAS functionality to replicate the schema without copying the data, which is exactly what the question asks for.

Question # 143

Date: October 27, 2024

A data engineer needs to create an Amazon Athena table based on a subset of data from an existing Athena table named cities_world. The cities_world table contains cities that are located around the world. The data engineer must create a new table named cities_us to contain only the cities from cities_world that are located in the US.

Which SQL statement should the data engineer use to meet this requirement?

A. INSERT INTO cities_usa (city,state) SELECT city, state FROM cities_world WHERE country='usa';
B. MOVE city, state FROM cities_world TO cities_usa WHERE country='usa';
C. INSERT INTO cities_usa SELECT city, state FROM cities_world WHERE country='usa';
D. UPDATE cities_usa SET (city, state) = (SELECT city, state FROM cities_world WHERE country='usa');

Correct Answer:

A. INSERT INTO cities_usa (city,state) SELECT city, state FROM cities_world WHERE country='usa';

B. MOVE city, state FROM cities_world TO cities_usa WHERE country='usa';

C. INSERT INTO cities_usa SELECT city, state FROM cities_world WHERE country='usa';

D. UPDATE cities_usa SET (city, state) = (SELECT city, state FROM cities_world WHERE country='usa');

Explanation:

The task requires creating a new table in Amazon Athena named `cities_us` that only includes cities from the existing `cities_world` table where the cities are located in the United States. The correct approach involves selecting data from the `cities_world` table and inserting it into the new `cities_us` table. Let's break down the options and why Option A is correct:

Option A: INSERT INTO cities_usa (city,state) SELECT city, state FROM cities_world WHERE country=’usa’;

- This statement is correct because it uses the `INSERT INTO` syntax, which is appropriate for adding data into an existing table. - The `SELECT` statement effectively filters the data from `cities_world` by selecting only the records where the `country` is 'usa'. - The fields `city` and `state` are specified explicitly, ensuring that the correct columns are inserted into `cities_usa`. - This option assumes that the `cities_us` table already exists and matches the structure expected by the `INSERT INTO` statement.

Option B: MOVE city, state FROM cities_world TO cities_usa WHERE country=’usa’;

- This syntax is incorrect for SQL and Amazon Athena. There is no `MOVE` statement in standard SQL or Athena to transfer data from one table to another.

Option C: INSERT INTO cities_usa SELECT city, state FROM cities_world WHERE country=’usa’;

- This option is similar to Option A but does not specify the columns `city` and `state` explicitly in the `INSERT INTO` clause. While in some SQL environments, this might work if the table structures exactly match, it is generally good practice to specify the columns explicitly as in Option A to avoid ambiguity and potential errors.

Option D: UPDATE cities_usa SET (city, state) = (SELECT city, state FROM cities_world WHERE country=’usa’);

- This option attempts to use an `UPDATE` statement, which is inappropriate for inserting new data into a table. `UPDATE` is used to modify existing records, not to insert new ones. - Furthermore, this syntax is incorrect for SQL because the `SET` clause is not used in this manner.

In summary, Option A is the only correct and syntactically appropriate choice for inserting data into an existing table in Amazon Athena, filtering the data based on a condition, and explicitly listing the columns involved.

Question # 144

Date: October 27, 2024

A company implements a data mesh that has a central governance account. The company needs to catalog all data in the governance account. The governance account uses AWS Lake Formation to centrally share data and grant access permissions.

The company has created a new data product that includes a group of Amazon Redshift Serverless tables. A data engineer needs to share the data product with a marketing team. The marketing team must have access to only a subset of columns. The data engineer needs to share the same data product with a compliance team. The compliance team must have access to a different subset of columns than the marketing team needs access to.

Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

A. Create views of the tables that need to be shared. Include only the required columns.
B. Create an Amazon Redshift data share that includes the tables that need to be shared.
C. Create an Amazon Redshift managed VPC endpoint in the marketing team's account. Grant the marketing team access to the views.
D. Share the Amazon Redshift data share to the Lake Formation catalog in the governance account.
E. Share the Amazon Redshift data share to the Amazon Redshift Serverless workgroup in the marketing team's account.

Correct Answer:

A. Create views of the tables that need to be shared. Include only the required columns.

B. Create an Amazon Redshift data share that includes the tables that need to be shared.

C. Create an Amazon Redshift managed VPC endpoint in the marketing team's account. Grant the marketing team access to the views.

D. Share the Amazon Redshift data share to the Lake Formation catalog in the governance account.

E. Share the Amazon Redshift data share to the Amazon Redshift Serverless workgroup in the marketing team's account.

Explanation:

To understand why options B and D are the correct choices, let's break down the requirements and the functionalities involved:

The scenario involves sharing specific subsets of data from Amazon Redshift Serverless tables with two different teams, each requiring access to different columns. The company uses AWS Lake Formation for central data governance and sharing.

1. Option B: Create an Amazon Redshift data share that includes the tables that need to be shared.

Amazon Redshift data sharing allows you to share data at a granular level between different Redshift clusters or even with other AWS services, without the need to duplicate data. By creating a data share, the data engineer can specify which tables, schemas, or even specific columns should be included. This allows the data engineer to prepare a data share that contains only the necessary data for each team, enabling focused, secure sharing of the data subsets required by the marketing and compliance teams.

2. Option D: Share the Amazon Redshift data share to the Lake Formation catalog in the governance account.

AWS Lake Formation is used here for centralized governance and access management. By sharing the Redshift data share with the Lake Formation catalog, the governance account can manage and enforce access permissions centrally. Lake Formation will then handle the sharing of this data with the respective teams, ensuring that each team only accesses the data they are authorized to see. This setup aligns with the company's requirement to use Lake Formation for centralized sharing and permissions, maintaining a controlled and compliant environment.

Now, let's briefly address why the other options are not suitable:

- Option A: Create views of the tables that need to be shared. Include only the required columns.

While creating views is a common method for restricting access to specific columns, it does not directly fulfill the requirement to centrally share and manage permissions through Lake Formation. Views do not inherently manage access or sharing across accounts; they are more suited for internal use within the same Redshift cluster.

- Option C: Create an Amazon Redshift managed VPC endpoint in the marketing team's account. Grant the marketing team access to the views.

This option involves network configuration and access control at a VPC level, which is not directly related to column-level access governance. It can complicate the sharing process and does not utilize Lake Formation's centralized governance capabilities.

- Option E: Share the Amazon Redshift data share to the Amazon Redshift Serverless workgroup in the marketing team's account.

While this approach could facilitate sharing, it does not incorporate Lake Formation for centralized governance, which is a key requirement in the scenario. Sharing directly with a Redshift Serverless workgroup bypasses the centralized control and auditing provided by Lake Formation.

In summary, options B and D effectively use Amazon Redshift's data sharing capabilities along with AWS Lake Formation's centralized governance to meet the company's data sharing and compliance requirements.

Question # 145

Date: October 27, 2024

A company has three subsidiaries. Each subsidiary uses a different data warehousing solution. The first subsidiary hosts its data warehouse in Amazon Redshift. The second subsidiary uses Teradata Vantage on AWS. The third subsidiary uses Google BigQuery.

The company wants to aggregate all the data into a central Amazon S3 data lake. The company wants to use Apache Iceberg as the table format.

A data engineer needs to build a new pipeline to connect to all the data sources, run transformations by using each source engine, join the data, and write the data to Iceberg.

Which solution will meet these requirements with the LEAST operational effort?

A. Use native Amazon Redshift, Teradata, and BigQuery connectors to build the pipeline in AWS Glue. Use native AWS Glue transforms to join the data. Run a Merge operation on the data lake Iceberg table.
B. Use the Amazon Athena federated query connectors for Amazon Redshift, Teradata, and BigQuery to build the pipeline in Athena. Write a SQL query to read from all the data sources, join the data, and run a Merge operation on the data lake Iceberg table.
C. Use the native Amazon Redshift connector, the Java Database Connectivity (JDBC) connector for Teradata, and the open source Apache Spark BigQuery connector to build the pipeline in Amazon EMR. Write code in PySpark to join the data. Run a Merge operation on the data lake Iceberg table.
D. Use the native Amazon Redshift, Teradata, and BigQuery connectors in Amazon Appflow to write data to Amazon S3 and AWS Glue Data Catalog. Use Amazon Athena to join the data. Run a Merge operation on the data lake Iceberg table.

Correct Answer:

A. Use native Amazon Redshift, Teradata, and BigQuery connectors to build the pipeline in AWS Glue. Use native AWS Glue transforms to join the data. Run a Merge operation on the data lake Iceberg table.

B. Use the Amazon Athena federated query connectors for Amazon Redshift, Teradata, and BigQuery to build the pipeline in Athena. Write a SQL query to read from all the data sources, join the data, and run a Merge operation on the data lake Iceberg table.

C. Use the native Amazon Redshift connector, the Java Database Connectivity (JDBC) connector for Teradata, and the open source Apache Spark BigQuery connector to build the pipeline in Amazon EMR. Write code in PySpark to join the data. Run a Merge operation on the data lake Iceberg table.

D. Use the native Amazon Redshift, Teradata, and BigQuery connectors in Amazon Appflow to write data to Amazon S3 and AWS Glue Data Catalog. Use Amazon Athena to join the data. Run a Merge operation on the data lake Iceberg table.

Explanation:

The correct answer is B, and here's why:

1. Amazon Athena Federated Query: Athena is a serverless query service that allows you to query data directly in Amazon S3 using standard SQL. One of its powerful features is the ability to perform federated queries, which means you can query data across multiple data sources without needing to extract and load the data into a single location first. This makes it suitable for scenarios where you want to aggregate data from different systems like Amazon Redshift, Teradata, and Google BigQuery.

2. Least Operational Effort: Using Athena federated queries minimizes operational complexity because it eliminates the need to manage separate ETL processes or data pipelines for each data source. By using federated connectors, you can directly query the data in its native environment. This reduces the need for data movement, simplifies the architecture, and decreases the operational overhead of maintaining separate connectors and transformation scripts for each data source.

3. SQL for Data Transformation: Since Athena supports SQL, you can write a single SQL query to read from all the data sources, apply any necessary transformations, join the data, and perform a Merge operation. This leverages existing SQL skills and streamlines the process of data transformation and integration.

4. Apache Iceberg Integration: Athena supports writing to Apache Iceberg tables. By using Athena, you can write the transformed and joined data directly into an Iceberg table format on Amazon S3. This integration is seamless and fits well with the requirement to use Iceberg as the table format for the data lake.

5. Serverless and Scalable: Athena is a serverless solution, which means there is no need to provision or manage infrastructure. It automatically scales with the query load, providing a cost-effective solution that can handle varying workloads without manual intervention.

In summary, option B leverages Amazon Athena's federated query capability to directly connect to and query multiple data sources using SQL, minimizing data movement and operational complexity. It allows the company to efficiently integrate data from disparate systems into a central data lake on Amazon S3 with Iceberg tables, fulfilling the requirements with minimal operational effort.

Question # 146

Date: November 06, 2024

A company stores customer data that contains personally identifiable information (PII) in an Amazon Redshift cluster. The company's marketing, claims, and analytics teams need to be able to access the customer data.

The marketing team should have access to obfuscated claim information but should have full access to customer contact information. The claims team should have access to customer information for each claim that the team processes. The analytics team should have access only to obfuscated PII data.

Which solution will enforce these data access requirements with the LEAST administrative overhead?

A. Create a separate Redshift cluster for each team. Load only the required data for each team. Restrict access to clusters based on the teams.
B. Create views that include required fields for each of the data requirements. Grant the teams access only to the view that each team requires.
C. Create a separate Amazon Redshift database role for each team. Define masking policies that apply for each team separately. Attach appropriate masking policies to each team role.
D. Move the customer data to an Amazon S3 bucket. Use AWS Lake Formation to create a data lake. Use fine-grained security capabilities to grant each team appropriate permissions to access the data.

Correct Answer:

A. Create a separate Redshift cluster for each team. Load only the required data for each team. Restrict access to clusters based on the teams.

B. Create views that include required fields for each of the data requirements. Grant the teams access only to the view that each team requires.

C. Create a separate Amazon Redshift database role for each team. Define masking policies that apply for each team separately. Attach appropriate masking policies to each team role.

D. Move the customer data to an Amazon S3 bucket. Use AWS Lake Formation to create a data lake. Use fine-grained security capabilities to grant each team appropriate permissions to access the data.

Explanation:

The correct answer, option C, involves creating separate Amazon Redshift database roles for each team and defining masking policies tailored to each team's access needs. This approach effectively addresses the company's requirements for controlled access while minimizing administrative overhead. Here's why this is the best solution:

1. Role-Based Access Control: By creating separate database roles for each team, you can easily manage and customize the access permissions specifically for the marketing, claims, and analytics teams. This allows for precise control over who can access what data, aligning perfectly with the need for differentiated data access.

2. Data Masking: Data masking policies in Amazon Redshift enable you to obfuscate sensitive information like PII. This is crucial for the marketing and analytics teams, who require access to obfuscated data to ensure privacy and compliance. By applying these policies to specific roles, you ensure that only the necessary data is visible to each team, reducing the risk of data exposure.

3. Reduced Administrative Overhead: Managing access through roles and masking policies is more straightforward and less labor-intensive than other options, like setting up separate clusters (option A) or creating and maintaining multiple views (option B). It centralizes the data management to a single Redshift cluster, allowing for easier scaling and maintenance.

4. Scalability and Flexibility: This approach allows for changes and updates to access policies without significant restructuring. If team roles or requirements change, adjustments can be made at the role or policy level, which is both flexible and efficient.

5. Security and Compliance: With role-based access and masking policies, you are ensuring that sensitive data is handled according to best practices, which is critical for compliance with data protection regulations that govern PII.

In contrast, option A involves significant duplication of data and infrastructure, leading to high costs and complexity. Option B, while viable, involves creating multiple views which can become cumbersome to manage as data requirements evolve. Option D, while leveraging AWS Lake Formation's fine-grained security, introduces additional complexity and potential latency by moving data to Amazon S3, which may not be necessary if the data already resides efficiently in Redshift.

Overall, option C provides a balanced, efficient, and secure way to meet the company's data access requirements with the least administrative burden.

Question # 147

Date: November 05, 2024

A gaming company uses Amazon Kinesis Data Streams to collect clickstream data. The company uses Amazon Data Firehose delivery streams to store the data in JSON format in Amazon S3. Data scientists at the company use Amazon Athena to query the most recent data to obtain business insights.

The company wants to reduce Athena costs but does not want to recreate the data pipeline.

Which solution will meet these requirements with the LEAST management effort?

A. Change the Firehose output format to Apache Parquet. Provide a custom S3 object YYYYMMDD prefix expression and specify a large buffer size. For the existing data, create an AWS Glue extract, transform, and load (ETL) job. Configure the ETL job to combine small JSON files, convert the JSON files to large Parquet files, and add the YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.
B. Create an Apache Spark job that combines JSON files and converts the JSON files to Apache Parquet files. Launch an Amazon EMR ephemeral cluster every day to run the Spark job to create new Parquet files in a different S3 location. Use the ALTER TABLE SET LOCATION statement to reflect the new S3 location on the existing Athena table.
C. Create a Kinesis data stream as a delivery destination for Firehose. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to run Apache Flink on the Kinesis data stream. Use Flink to aggregate the data and save the data to Amazon S3 in Apache Parquet format with a custom S3 object YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.
D. Integrate an AWS Lambda function with Firehose to convert source records to Apache Parquet and write them to Amazon S3. In parallel, run an AWS Glue extract, transform, and load (ETL) job to combine the JSON files and convert the JSON files to large Parquet files. Create a custom S3 object YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

Correct Answer:

A. Change the Firehose output format to Apache Parquet. Provide a custom S3 object YYYYMMDD prefix expression and specify a large buffer size. For the existing data, create an AWS Glue extract, transform, and load (ETL) job. Configure the ETL job to combine small JSON files, convert the JSON files to large Parquet files, and add the YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

B. Create an Apache Spark job that combines JSON files and converts the JSON files to Apache Parquet files. Launch an Amazon EMR ephemeral cluster every day to run the Spark job to create new Parquet files in a different S3 location. Use the ALTER TABLE SET LOCATION statement to reflect the new S3 location on the existing Athena table.

C. Create a Kinesis data stream as a delivery destination for Firehose. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to run Apache Flink on the Kinesis data stream. Use Flink to aggregate the data and save the data to Amazon S3 in Apache Parquet format with a custom S3 object YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

D. Integrate an AWS Lambda function with Firehose to convert source records to Apache Parquet and write them to Amazon S3. In parallel, run an AWS Glue extract, transform, and load (ETL) job to combine the JSON files and convert the JSON files to large Parquet files. Create a custom S3 object YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

Explanation:

The correct solution to reduce Amazon Athena costs, without recreating the existing data pipeline, involves optimizing how the data is stored in Amazon S3. The key here is to focus on reducing the amount of data scanned by Athena queries, which directly influences cost. Athena charges are based on the amount of data scanned during queries, so reducing the file size and optimizing the data format can significantly cut costs.

Here's a detailed explanation of the options:

A. Change the Firehose output format to Apache Parquet:

- Apache Parquet is a columnar storage file format that is highly efficient for both storage and query performance. It allows for more efficient data compression and encoding schemes, which reduces the amount of data scanned by Athena. - Custom S3 object YYYYMMDD prefix: This helps in organizing the data and can improve query performance by allowing partitioning based on time. This way, Athena can skip scanning data outside the relevant date range, further reducing costs. - Large buffer size: This reduces the frequency of small writes to S3, which can create many small files. Fewer, larger files are more efficient for Athena to process. - AWS Glue ETL job for existing data: This job would transform existing JSON files into the Parquet format, thus optimizing historical data for Athena queries. - ALTER TABLE ADD PARTITION: This ensures that the new data structure is recognized by Athena, allowing it to take advantage of the partitioning for efficient querying.

B. Create an Apache Spark job on Amazon EMR: Launching an ephemeral cluster daily involves more management overhead than desired. This option is more complex due to managing EMR clusters and scheduling jobs.

C. Use Amazon Managed Service for Apache Flink: This involves a more complex setup and additional management of Kinesis data streams and Flink applications, which increases the management effort compared to simply changing the output format in Firehose and using Glue.

D. Integrate AWS Lambda with Firehose: While converting records to Parquet with Lambda is possible, running a parallel AWS Glue ETL job adds complexity. Moreover, managing two processes (Lambda for real-time and Glue for batch processing) can increase the management effort.

Option A offers the least management effort by leveraging existing AWS services (Firehose and Glue) with minimal changes to the existing data pipeline. It allows the company to achieve cost-effective storage and querying in Athena, while avoiding the need for complex additional infrastructure or substantial changes in data flow.

Question # 148

Date: October 27, 2024

A company uses Amazon S3 to store data and Amazon QuickSight to create visualizations,

The company has an S3 bucket in an AWS account named Hub-Account. The S3 bucket is encrypted by an AWS Key Management Service (AWS KMS) key. The company's QuickSight instance is in a separate account named BI-Account.

The company updates the S3 bucket policy to grant access to the QuickSight service role. The company wants to enable cross-account access to allow QuickSight to interact with the S3 bucket.

Which combination of steps will meet this requirement? (Choose two.)

A. Use the existing AWS KMS key to encrypt connections from QuickSight to the S3 bucket.
B. Add the S3 bucket as a resource that the QuickSight service role can access.
C. Use AWS Resource Access Manager (AWS RAM) to share the S3 bucket with the BI-Account account.
D. Add an IAM policy to the QuickSight service role to give QuickSight access to the KMS key that encrypts the S3 bucket.
E. Add the KMS key as a resource that the QuickSight service role can access.

Correct Answer:

A. Use the existing AWS KMS key to encrypt connections from QuickSight to the S3 bucket.

B. Add the S3 bucket as a resource that the QuickSight service role can access.

C. Use AWS Resource Access Manager (AWS RAM) to share the S3 bucket with the BI-Account account.

D. Add an IAM policy to the QuickSight service role to give QuickSight access to the KMS key that encrypts the S3 bucket.

E. Add the KMS key as a resource that the QuickSight service role can access.

Explanation:

The question revolves around enabling cross-account access between Amazon QuickSight, which resides in one AWS account (BI-Account), and an Amazon S3 bucket encrypted with an AWS KMS key in another AWS account (Hub-Account). To achieve this, you need to ensure that both the S3 bucket and the KMS key are accessible from the QuickSight service role in the BI-Account.

Let’s break down the options:

A. Use the existing AWS KMS key to encrypt connections from QuickSight to the S3 bucket. - This option doesn't directly address the requirement of granting access to the resources from another account. The encryption key's purpose is to secure data, not to manage access permissions.

B. Add the S3 bucket as a resource that the QuickSight service role can access. - This is a necessary step. By updating the S3 bucket policy, you can allow the QuickSight service role in the BI-Account to access the S3 bucket in the Hub-Account. This involves specifically listing the QuickSight service role as a trusted entity that can perform actions on the S3 bucket.

C. Use AWS Resource Access Manager (AWS RAM) to share the S3 bucket with the BI-Account account. - AWS RAM is not typically used for sharing S3 buckets. It's more commonly applied for sharing resources like VPCs, subnets, and other network-related resources. This option is not applicable for the scenario described.

D. Add an IAM policy to the QuickSight service role to give QuickSight access to the KMS key that encrypts the S3 bucket. - This is crucial because, for QuickSight to read data from the S3 bucket, it must decrypt the data encrypted with the KMS key. The QuickSight service role in the BI-Account needs the necessary permissions on the KMS key that encrypts the S3 data to perform decryption operations.

E. Add the KMS key as a resource that the QuickSight service role can access. - This option is aligned with allowing the QuickSight service to use the KMS key for decryption. It ensures that the necessary permissions are in place for the QuickSight role to interact with the encrypted data.

To summarize, the correct steps involve: 1. Updating the S3 bucket policy to permit access by the QuickSight service role (Option B). 2. Ensuring that the QuickSight service role has the necessary permissions on the KMS key to decrypt the data (Option D or E).

These steps will allow the required cross-account access for QuickSight to interact with the encrypted S3 bucket data.

Question # 149

Date: November 04, 2024

A company uses Amazon Redshift as its data warehouse. Data encoding is applied to the existing tables of the data warehouse. A data engineer discovers that the compression encoding applied to some of the tables is not the best fit for the data.

The data engineer needs to improve the data encoding for the tables that have sub-optimal encoding.

Which solution will meet this requirement?

A. Run the ANALYZE command against the identified tables. Manually update the compression encoding of columns based on the output of the command.
B. Run the ANALYZE COMPRESSION command against the identified tables. Manually update the compression encoding of columns based on the output of the command.
C. Run the VACUUM REINDEX command against the identified tables.
D. Run the VACUUM RECLUSTER command against the identified tables.

Correct Answer:

A. Run the ANALYZE command against the identified tables. Manually update the compression encoding of columns based on the output of the command.

B. Run the ANALYZE COMPRESSION command against the identified tables. Manually update the compression encoding of columns based on the output of the command.

C. Run the VACUUM REINDEX command against the identified tables.

D. Run the VACUUM RECLUSTER command against the identified tables.

Explanation:

The correct answer is B: Run the ANALYZE COMPRESSION command against the identified tables. Manually update the compression encoding of columns based on the output of the command.

Here's why this is the correct solution:

1. Purpose of ANALYZE COMPRESSION: The ANALYZE COMPRESSION command in Amazon Redshift is specifically designed to analyze existing data and suggest the best compression encodings for each column in a table. This is important because optimal compression can significantly reduce storage requirements and improve query performance by minimizing the amount of data that needs to be read from disk.

2. Identifying Sub-optimal Encoding: When a data engineer discovers that the current compression encoding is not optimal, the first step is to identify which columns could benefit from a different encoding. The ANALYZE COMPRESSION command provides a detailed report indicating the current encoding and suggests the most efficient encoding based on the actual data in each column.

3. Manual Update Based on Output: After running the ANALYZE COMPRESSION command, the data engineer will receive output that includes recommendations for the best compression encodings. The engineer can then use this information to manually alter the table and update the column encodings to the recommended settings. This manual step is necessary because Redshift does not automatically change the compression; it requires manual intervention to apply the suggested changes.

4. Why Other Options Are Incorrect: - Option A (ANALYZE command) is incorrect because the ANALYZE command is used to update statistics for query planning, not for suggesting compression encodings. - Option C (VACUUM REINDEX) is incorrect because VACUUM REINDEX is used to re-index tables and does not relate to compression settings. - Option D (VACUUM RECLUSTER) is incorrect because VACUUM RECLUSTER is used to reorganize the data distribution for tables, especially those with interleaved sort keys, and does not affect compression encoding.

In summary, the ANALYZE COMPRESSION command is tailored for assessing and optimizing column compression encodings, making it the right choice for improving data encoding in Amazon Redshift tables.

Question # 150

Date: November 04, 2024

A retail company stores customer data in an Amazon S3 bucket. Some of the customer data contains personally identifiable information (PII) about customers. The company must not share PII data with business partners.

A data engineer must determine whether a dataset contains PII before making objects in the dataset available to business partners.

Which solution will meet this requirement with the LEAST manual intervention?

A. Configure the S3 bucket and S3 objects to allow access to Amazon Macie. Use automated sensitive data discovery in Macie.
B. Configure AWS CloudTrail to monitor S3 PUT operations. Inspect the CloudTrail trails to identify operations that save PII.
C. Create an AWS Lambda function to identify PII in S3 objects. Schedule the function to run periodically.

D. Create a table in AWS Glue Data Catalog. Write custom SQL queries to identify PII in the table. Use Amazon Athena to run the queries.

Correct Answer:

A. Configure the S3 bucket and S3 objects to allow access to Amazon Macie. Use automated sensitive data discovery in Macie.

B. Configure AWS CloudTrail to monitor S3 PUT operations. Inspect the CloudTrail trails to identify operations that save PII.

C. Create an AWS Lambda function to identify PII in S3 objects. Schedule the function to run periodically.

D. Create a table in AWS Glue Data Catalog. Write custom SQL queries to identify PII in the table. Use Amazon Athena to run the queries.

Explanation:

It looks like there is an error in the answer provided as "nan," which typically stands for "not a number" and is not a valid option. However, I can provide an explanation for why Option A is likely the best choice for solving the problem with the least manual intervention, based on the context of the question.

Option A suggests configuring the S3 bucket and S3 objects to allow access to Amazon Macie and using its automated sensitive data discovery feature. Here's why this is the most suitable solution:

1. Purpose-built for PII Detection: Amazon Macie is a fully managed data security and data privacy service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. It is specifically designed to identify and alert users about sensitive data, such as personally identifiable information (PII), stored in Amazon S3.

2. Automation: Macie's automated sensitive data discovery feature means that it can continuously and automatically inspect the data in S3 to detect sensitive information with minimal manual intervention. This makes it ideal for ensuring that any PII is identified before data is shared with business partners.

3. Scalability and Maintenance: Since Macie is a managed service, AWS handles the scaling and maintenance aspects. This reduces the need for ongoing manual maintenance and allows the data engineering team to focus on other tasks without worrying about the infrastructure or scaling issues.

4. Ease of Integration: Configuring S3 to work with Macie is straightforward, and once set up, Macie continuously monitors the data for sensitive information. This integration minimizes the need for custom coding or complex configuration.

In contrast, the other options (B, C, and D) involve more manual processes or custom development:

- Option B requires manual inspection of CloudTrail logs, which can be labor-intensive and error-prone. - Option C involves creating and maintaining a custom Lambda function, which adds complexity and ongoing maintenance tasks. - Option D requires setting up a data catalog and writing custom SQL queries, which also involves significant manual effort and ongoing management.

Therefore, Option A, utilizing Amazon Macie's automated capabilities, is likely the best approach to meet the requirement of identifying PII with the least manual intervention.

Question # 151

Date: October 31, 2024

A company has a data lake in Amazon S3. The company collects AWS CloudTrail logs for multiple applications. The company stores the logs in the data lake, catalogs the logs in AWS Glue, and partitions the logs based on the year. The company uses Amazon Athena to analyze the logs.

Recently, customers reported that a query on one of the Athena tables did not return any data. A data engineer must resolve the issue.

Which combination of troubleshooting steps should the data engineer take? (Choose two.)

A. Confirm that Athena is pointing to the correct Amazon S3 location.
B. Increase the query timeout duration.
C. Use the MSCK REPAIR TABLE command.
D. Restart Athena.
E. Delete and recreate the problematic Athena table.

Correct Answer:

A. Confirm that Athena is pointing to the correct Amazon S3 location.

B. Increase the query timeout duration.

C. Use the MSCK REPAIR TABLE command.

D. Restart Athena.

E. Delete and recreate the problematic Athena table.

Explanation:

It seems there is an issue with the given answer as it is listed as "nan," which isn't a valid answer. However, based on the provided options and the typical troubleshooting steps for resolving issues with Athena queries and AWS Glue catalog, I can provide an explanation for which options would likely be correct:

1. Confirm that Athena is pointing to the correct Amazon S3 location (Option A): - This is an important step because Athena queries data directly from Amazon S3 using the location specified in the table's metadata. If the location is incorrect or has changed, Athena will not be able to find the data it needs to query. Checking and confirming the correct S3 path ensures that Athena is looking in the right place for the logs.

2. Use the MSCK REPAIR TABLE command (Option C): - In AWS Glue and Athena, if new partitions are added to a table, they may not automatically be recognized by the system. The MSCK REPAIR TABLE command is used to update the metadata catalog with the current partitions in the S3 bucket. Running this command can help resolve issues where queries return no data because the partition information is outdated or incomplete.

The other options, such as increasing the query timeout duration, restarting Athena, or deleting and recreating the problematic table, do not directly address the typical root causes of no data being returned in Athena queries related to S3 location misconfiguration or partition metadata issues. These options are less likely to resolve the issue given the context of the problem.

Question # 152

Date: November 04, 2024

A data engineer maintains custom Python scripts that perform a data formatting process that many AWS Lambda functions use. When the data engineer needs to modify the Python scripts, the data engineer must manually update all the Lambda functions.

The data engineer requires a less manual way to update the Lambda functions.

Which solution will meet this requirement?

A. Store the custom Python scripts in a shared Amazon S3 bucket. Store a pointer to the custom scripts in the execution context object.
B. Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions.
C. Store the custom Python scripts in a shared Amazon S3 bucket. Store a pointer to the customer scripts in environment variables.
D. Assign the same alias to each Lambda function. Call each Lambda function by specifying the function's alias.

Correct Answer:

A. Store the custom Python scripts in a shared Amazon S3 bucket. Store a pointer to the custom scripts in the execution context object.

B. Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions.

C. Store the custom Python scripts in a shared Amazon S3 bucket. Store a pointer to the customer scripts in environment variables.

D. Assign the same alias to each Lambda function. Call each Lambda function by specifying the function's alias.

Explanation:

The correct answer is B: Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions.

Here's why this solution is ideal:

1. Understanding Lambda Layers: AWS Lambda layers allow you to package additional code, such as libraries or custom scripts, separately from your main deployment package. This makes it easier to manage dependencies and makes the main deployment package smaller and more efficient. By using layers, you can share common code across multiple Lambda functions without duplicating it in each function's deployment package.

2. Separation of Concerns: By packaging the custom Python scripts into a Lambda layer, you're separating the scripts from the individual Lambda functions. This means that if you need to update the scripts, you only need to update the layer, and all Lambda functions that use this layer will automatically have access to the updated code.

3. Ease of Updates: With Lambda layers, updating the shared scripts becomes much easier. You simply update the layer with the new version of the scripts, and all the Lambda functions that use this layer will use the updated scripts without any further changes to those functions.

4. Consistency and Reusability: Using layers ensures that all Lambda functions utilize the exact same version of the scripts, reducing the risk of inconsistencies. This also promotes code reuse and simplifies maintenance since you have a single point of update for the scripts.

5. Efficient Resource Management: Layers can be shared across different Lambda functions and even across different AWS accounts, which makes them a scalable and efficient way to manage shared code resources.

Options A and C involve storing the scripts in an S3 bucket and using pointers in the execution context or environment variables. While this method could technically work, it introduces additional complexity and potential delays since each Lambda function would need to retrieve the scripts from S3 every time it runs. This is less efficient and more error-prone than using layers.

Option D suggests using aliases, which are primarily used for version control and deployment management within Lambda rather than managing shared dependencies like scripts.

In summary, packaging the Python scripts into Lambda layers and applying them to the necessary functions provides a streamlined, maintainable, and scalable solution to manage shared code across multiple AWS Lambda functions.

Question # 153

Date: November 04, 2024

A company stores customer data in an Amazon S3 bucket. Multiple teams in the company want to use the customer data for downstream analysis. The company needs to ensure that the teams do not have access to personally identifiable information (PII) about the customers.

Which solution will meet this requirement with LEAST operational overhead?

A. Use Amazon Macie to create and run a sensitive data discovery job to detect and remove PII.
B. Use S3 Object Lambda to access the data, and use Amazon Comprehend to detect and remove PII.
C. Use Amazon Data Firehose and Amazon Comprehend to detect and remove PII.
D. Use an AWS Glue DataBrew job to store the PII data in a second S3 bucket. Perform analysis on the data that remains in the original S3 bucket.

Correct Answer:

A. Use Amazon Macie to create and run a sensitive data discovery job to detect and remove PII.

B. Use S3 Object Lambda to access the data, and use Amazon Comprehend to detect and remove PII.

C. Use Amazon Data Firehose and Amazon Comprehend to detect and remove PII.

D. Use an AWS Glue DataBrew job to store the PII data in a second S3 bucket. Perform analysis on the data that remains in the original S3 bucket.

Explanation:

The question asks for a solution that allows multiple teams to access customer data stored in an Amazon S3 bucket for analysis, but without access to personally identifiable information (PII). Additionally, the solution should have the least operational overhead.

Let's evaluate the options provided:

A. Use Amazon Macie to create and run a sensitive data discovery job to detect and remove PII. Amazon Macie is a fully managed data security and privacy service that uses machine learning to discover and protect sensitive data in AWS. It can automatically detect PII in S3 buckets. However, using Macie for regular data processing and transformation might involve setting up additional workflows for data access, which could lead to some operational overhead.

B. Use S3 Object Lambda to access the data, and use Amazon Comprehend to detect and remove PII. S3 Object Lambda allows you to process and transform data as it is retrieved from S3 without needing to store the transformed data. Amazon Comprehend is capable of detecting PII, but integrating these services might require custom development and management, resulting in operational overhead.

C. Use Amazon Data Firehose and Amazon Comprehend to detect and remove PII. Amazon Kinesis Data Firehose can ingest and transform streaming data, but it's more suited to real-time data processing rather than static data analysis. Using it with Amazon Comprehend for PII detection might not be the most straightforward solution for data already stored in S3, potentially leading to higher complexity and overhead.

D. Use an AWS Glue DataBrew job to store the PII data in a second S3 bucket. Perform analysis on the data that remains in the original S3 bucket. AWS Glue DataBrew is a data preparation service that allows you to clean and normalize data with visual tools. You can use it to detect and mask PII within a dataset. By setting up a DataBrew job to handle this, you can automate the process of separating PII from non-PII data, ensuring the teams access only the sanitized data. This approach is efficient in terms of operational overhead because it leverages a service specifically designed for data preparation and transformation without requiring extensive custom development or ongoing manual intervention.

Given these considerations, the solution with the least operational overhead that effectively meets the requirement is option D. AWS Glue DataBrew is well-suited for this task as it allows for automated data preparation and transformation, ensuring that teams can access the necessary data without exposure to PII.

Question # 154

Date: November 04, 2024

A company stores its processed data in an S3 bucket. The company has a strict data access policy. The company uses IAM roles to grant teams within the company different levels of access to the S3 bucket.

The company wants to receive notifications when a user violates the data access policy. Each notification must include the username of the user who violated the policy.

Which solution will meet these requirements?

A. Use AWS Config rules to detect violations of the data access policy. Set up compliance alarms.
B. Use Amazon CloudWatch metrics to gather object-level metrics. Set up CloudWatch alarms.
C. Use AWS CloudTrail to track object-level events for the S3 bucket. Forward events to Amazon CloudWatch to set up CloudWatch alarms.
D. Use Amazon S3 server access logs to monitor access to the bucket. Forward the access logs to an Amazon CloudWatch log group. Use metric filters on the log group to set up CloudWatch alarms.

Correct Answer:

A. Use AWS Config rules to detect violations of the data access policy. Set up compliance alarms.

B. Use Amazon CloudWatch metrics to gather object-level metrics. Set up CloudWatch alarms.

C. Use AWS CloudTrail to track object-level events for the S3 bucket. Forward events to Amazon CloudWatch to set up CloudWatch alarms.

D. Use Amazon S3 server access logs to monitor access to the bucket. Forward the access logs to an Amazon CloudWatch log group. Use metric filters on the log group to set up CloudWatch alarms.

Explanation:

The question asks for a solution that allows a company to receive notifications when a user violates a data access policy in an S3 bucket, and these notifications must include the username of the violator.

Let's break down the options:

A. AWS Config rules: AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. While it can monitor compliance with some predefined rules and configurations, it does not inherently track user-specific activities or provide real-time notifications with username details. Therefore, it is not suitable for tracking real-time access violations specifically related to user activities.

B. Amazon CloudWatch metrics: CloudWatch is primarily used for monitoring AWS resources and applications, providing actionable insights through alarms and dashboards. However, CloudWatch metrics alone do not provide detailed, user-specific logging of actions like accessing an S3 bucket. They are better suited for monitoring performance and operational metrics, not for capturing detailed access logs with usernames.

C. AWS CloudTrail: CloudTrail is designed to log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. It records user activity and API calls, providing detailed information about who did what and when. By logging object-level events for the S3 bucket, CloudTrail can capture and report activities, including the username of the user who accessed or attempted to access the bucket. These logs can then be forwarded to CloudWatch Logs, where you can set up alarms based on specific patterns, such as unauthorized access attempts. This approach meets the requirement of notifying the company of policy violations and includes the username in the notifications.

D. Amazon S3 server access logs: S3 server access logs provide detailed records for the requests made to an S3 bucket. While these logs can capture user-specific access details, including the username, they are not designed for real-time monitoring. They are more suitable for retrospective analysis. Furthermore, setting up metric filters and alarms based on these logs would be complex and not as seamless as using CloudTrail with CloudWatch.

Therefore, the most appropriate solution is option C, which leverages AWS CloudTrail to track object-level events and forward them to Amazon CloudWatch. This setup allows for the creation of alarms that can notify the company of access violations, including the username of the violator, thus fulfilling all the requirements mentioned in the question.

Question # 155

Date: November 04, 2024

A company needs to load customer data that comes from a third party into an Amazon Redshift data warehouse. The company stores order data and product data in the same data warehouse. The company wants to use the combined dataset to identify potential new customers.

A data engineer notices that one of the fields in the source data includes values that are in JSON format.

How should the data engineer load the JSON data into the data warehouse with the LEAST effort?

A. Use the SUPER data type to store the data in the Amazon Redshift table.
B. Use AWS Glue to flatten the JSON data and ingest it into the Amazon Redshift table.
C. Use Amazon S3 to store the JSON data. Use Amazon Athena to query the data.
D. Use an AWS Lambda function to flatten the JSON data. Store the data in Amazon S3.

Correct Answer:

A. Use the SUPER data type to store the data in the Amazon Redshift table.

B. Use AWS Glue to flatten the JSON data and ingest it into the Amazon Redshift table.

C. Use Amazon S3 to store the JSON data. Use Amazon Athena to query the data.

D. Use an AWS Lambda function to flatten the JSON data. Store the data in Amazon S3.

Explanation:

It seems there's a misunderstanding in the provided answer. The correct answer is missing, represented by "nan," which implies no answer has been selected. Let's analyze the options to determine the most suitable one for this scenario.

The goal is to load JSON data into an Amazon Redshift data warehouse with the least effort. The options provided include different methods for handling and storing JSON data. Here's a breakdown of each option:

A. Use the SUPER data type to store the data in the Amazon Redshift table. - The SUPER data type in Amazon Redshift is designed to natively store semi-structured data, including JSON, without requiring flattening. This option allows you to ingest JSON data directly into Redshift tables and then use Redshift's native functions to query and manipulate the data. This method involves minimal effort because it eliminates the need for data transformation outside Redshift and leverages built-in capabilities.

B. Use AWS Glue to flatten the JSON data and ingest it into the Amazon Redshift table. - AWS Glue is a fully managed ETL service that can be used to transform and load data. While it can flatten JSON data, setting up and maintaining Glue jobs involves more effort compared to directly using Redshift's capabilities. It requires creating and managing ETL processes, which adds complexity.

C. Use Amazon S3 to store the JSON data. Use Amazon Athena to query the data. - Storing JSON in Amazon S3 and querying it with Athena is a viable approach for analytics, but it doesn't directly load the data into Redshift. Instead, it keeps the data separate, which might not meet the requirement of having a combined dataset in Redshift for analysis.

D. Use an AWS Lambda function to flatten the JSON data. Store the data in Amazon S3. - Using Lambda to process JSON and store it in S3 also adds complexity. While it can be automated, it involves setting up Lambda functions and maintaining them. Similar to option C, it doesn't directly integrate the data into Redshift.

Considering these options, the most straightforward solution with the least effort is to use option A, the SUPER data type, in Amazon Redshift. This choice takes advantage of Redshift's ability to handle semi-structured data natively, streamlining the process of loading and querying JSON data without the need for additional transformation or external services.