Amazon

Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)

NEW QUESTION 1
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

Answer: B

Explanation:
Amazon AppFlow is a fully managed integration service that enables you to securely transfer data between SaaS applications and AWS services like Amazon S3 and AmazonRedshift. Amazon AppFlow supports many SaaS applications as data sources and targets, and allows you to configure data flows with a few clicks. Amazon AppFlow also provides features such as data transformation, filtering, validation, and encryption to prepare and protect your data. Amazon AppFlow meets the requirements of the media company with the least operational overhead, as it eliminates the need to write code, manage infrastructure, or monitor data pipelines. References:
? Amazon AppFlow
? Amazon AppFlow | SaaS Integrations List
? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

NEW QUESTION 2
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 layer
C. Apply the Lambda layers to the Lambda functions.
D. Store a pointer to the custom Python scripts in environment variables in a shared Amazon S3 bucket.
E. Assign the same alias to each Lambda functio
F. Call reach Lambda function by specifying the function's alias.

Answer: B

Explanation:
Lambda layers are a way to share code and dependencies across multiple Lambda functions. By packaging the custom Python scripts into Lambda layers, the data engineer can update the scripts in one place and have them automatically applied to all the Lambda functions that use the layer. This reduces the manual effort and ensures consistency across the Lambda functions. The other options are either not feasible or not efficient. Storing a pointer to the custom Python scripts in the execution context object or in environment variables would require the Lambda functions to download the scripts from Amazon S3 every time they are invoked, which would increase latency and cost. Assigning the same alias to each Lambda function would not help with updating the Python scripts, as the alias only points to a specific version of the Lambda function code. References:
? AWS Lambda layers
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.4: AWS Lambda

NEW QUESTION 3
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 classe
B. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classe
C. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.
D. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequentl
E. 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.
F. Use S3 Intelligent-Tierin
G. Activate the Deep Archive Access tier.
H. Use S3 Intelligent-Tierin
I. Use the default access tier.

Answer: D

Explanation:
S3 Intelligent-Tiering is a storage class that automatically moves objects between four access tiers based on the changing access patterns. The default access tier consists of two tiers: Frequent Access and Infrequent Access. Objects in the Frequent Access tier have the same performance and availability as S3 Standard, while objects in the Infrequent Access tier have the same performance and availability as S3 Standard-IA. S3 Intelligent-Tiering monitors the access patterns of each object and moves them between the tiers accordingly, without any operational overhead or retrieval fees. This solution can optimize S3 storage costs for data with unpredictable and variable access patterns, while ensuring millisecond latency for data retrieval. The other solutions are not optimal or relevant for this requirement. Using S3 Storage Lens standard metrics and activity metrics can provide insights into the storage usage and access patterns, but they do not automate the data movement between storage classes. Creating S3 Lifecycle policies for the S3 buckets can move objects to more cost-optimized storage classes, but they require manual configuration and maintenance, and they may incur retrieval fees for data that is accessed unexpectedly. Activating the Deep Archive Access tier for S3 Intelligent-Tiering can further reduce the storage costs for data that is rarely accessed, but it also increases the retrieval time to 12 hours, which does not meet the requirement of millisecond latency. References:
? S3 Intelligent-Tiering
? S3 Storage Lens

? S3 Lifecycle policies
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 4
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.

Answer: A

Explanation:
The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). References:
? Introducing AWS Glue Flex jobs: Cost savings on ETL workloads
? Serverless Data Integration – AWS Glue Pricing
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

NEW QUESTION 5
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.

Answer: A

Explanation:
This solution meets the requirements of managing the ingestion of real-time streaming data into AWS and performing real-time analytics on the incoming streaming data with the least operational overhead. Amazon Managed Service for Apache Flink is a fully managed service that allows you to run Apache Flink applications without having to manage any infrastructure or clusters. Apache Flink is a framework for stateful stream processing that supports various types of aggregations, such as tumbling, sliding, and session windows, over streaming data. By using Amazon Managed Service for Apache Flink, you can easily connect to Amazon Kinesis Data Streams as the source and sink of your streaming data, and perform time-based analytics over a window of up to 30 minutes. This solution is also highly fault tolerant, as Amazon Managed Service for Apache Flink automatically scales, monitors, and restarts your Flink applications in case of failures. References:
? Amazon Managed Service for Apache Flink
? Apache Flink
? Window Aggregations in Flink

NEW QUESTION 6
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 even/ hour.
B. Use AWS Glue DataBrewto clean and prepare the data for analytics.
C. Use AWS Lambda functions to schedule and run the ETL jobs even/ 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.

Answer: AD

Explanation:
The correct answer is to configure AWS Glue triggers to run the ETL jobs every hour and use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift. AWS Glue triggers are a way to schedule and orchestrate ETL jobs with the least operational overhead. AWS Glue connections are a way to securely connect to data sources and targets using JDBC or MongoDB drivers. AWS Glue DataBrew is a visual data preparation tool that does not support MongoDB as a data source. AWS Lambda functions are a serverless option to schedule and run ETL jobs, but they have a limit of 15 minutes for execution time, which may not be enough for complex transformations. The Redshift Data API is a way to run SQL commands on Amazon Redshift clusters without needing a persistent connection, but it does not support loading data from AWS Glue ETL jobs. References:
? AWS Glue triggers
? AWS Glue connections
? AWS Glue DataBrew
? [AWS Lambda functions]
? [Redshift Data API]

NEW QUESTION 7

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 QuickSgqht and Athena.
B. The Athena tables are not cataloged.
C. QuickSiqht does not have access to the S3 bucket.
D. QuickSight does not have access to decrypt S3 data.
E. There is no 1AM role assigned to QuickSiqht.

Answer: CD

Explanation:
QuickSight does not have access to the S3 bucket and QuickSight does not have access to decrypt S3 data are two possible factors that could cause the permissions- related errors. Amazon QuickSight is a business intelligence service that allows you to create and share interactive dashboards based on various data sources, including Amazon Athena. Amazon Athena is a serverless query service that allows you to analyze data stored in Amazon S3 using standard SQL. To use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket, you need to grant QuickSight access to both Athena and S3, as well as any encryption keys that are used to encrypt the S3 data. If QuickSight does not have access to the S3 bucket or the encryption keys, it will not be able to read the data from Athena and display it on the dashboard, resulting in an error message that indicates insufficient permissions.
The other options are not factors that could cause the permissions-related errors. Option A, there is no connection between QuickSight and Athena, is not a factor, as QuickSight supports Athena as a native data source, and you can easily create a connection between them using the QuickSight console or the API. Option B, the Athena tables are not cataloged, is not a factor, as QuickSight can automatically discover the Athena tables that are cataloged in the AWS Glue Data Catalog, and you can also manually specify the Athena tables that are not cataloged. Option E, there is no IAM role assigned to QuickSight, is not a factor, as QuickSight requires an IAM role to access any AWS data sources, including Athena and S3, and you can create and assign an IAM role to QuickSight using the QuickSight console or the API. References:
? Using Amazon Athena as a Data Source
? Granting Amazon QuickSight Access to AWS Resources
? Encrypting Data at Rest in Amazon S3

NEW QUESTION 8
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 dat
B. Store the data in Amazon S3 for querying.
C. Use AWS Lambda to process the sensor dat
D. Store the data in Amazon S3 for querying.
E. Use Amazon Kinesis Data Streams to capture the sensor dat
F. Store the data in Amazon DynamoDB for querying.
G. Use Amazon Simple Queue Service(Amazon SQS) to buffer incomingsensor dat
H. Use AWS Glue to store thedata in Amazon RDS for querying.

Answer: C

Explanation:
Amazon Kinesis Data Streams is a service that enables you to collect, process, and analyze streaming data in real time. You can use Kinesis Data Streams to capture sensor data from various sources, such as IoT devices, web applications, or mobile apps. You can create data streams that can scale up to handle any amount of data from thousands of producers. You can also use the Kinesis Client Library (KCL) or the Kinesis Data Streams API to write applications that process and analyze the data in the streams1. Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use DynamoDB to store the sensor data in nested JSON format, as DynamoDB supports document data types, such as lists and maps. You can also use DynamoDB to query the data with a latency of less than 10 milliseconds, as DynamoDB offers single-digit millisecond performance for any scale of data. You can use the DynamoDB API or the AWS SDKs to perform queries on the data, such as using key-value lookups, scans, or queries2.
The solution that meets the requirements with the least operational overhead is to use Amazon Kinesis Data Streams to capture the sensor data and store the data in Amazon DynamoDB for querying. This solution has the following advantages:
? It does not require you to provision, manage, or scale any servers, clusters, or queues, as Kinesis Data Streams and DynamoDB are fully managed services that handle all the infrastructure for you. This reduces the operational complexity and cost of running your solution.
? It allows you to ingest sensor data in near real time, as Kinesis Data Streams can capture data records as they are produced and deliver them to your applications within seconds. You can also use Kinesis Data Firehose to load the data from the streams to DynamoDB automatically and continuously3.
? It allows you to store the data in nested JSON format, as DynamoDB supports document data types, such as lists and maps. You can also use DynamoDB Streams to capturechanges in the data and trigger actions, such as sending notifications or updating other databases.
? It allows you to query the data with a latency of less than 10 milliseconds, as DynamoDB offers single-digit millisecond performance for any scale of data. You can also use DynamoDB Accelerator (DAX) to improve the read performance by caching frequently accessed data.
Option A is incorrect because it suggests using a self-hosted Apache Kafka cluster to capture the sensor data and store the data in Amazon S3 for querying. This solution has the following disadvantages:
? It requires you to provision, manage, and scale your own Kafka cluster, either on EC2 instances or on-premises servers. This increases the operational complexity and cost of running your solution.
? It does not allow you to query the data with a latency of less than 10 milliseconds, as Amazon S3 is an object storage service that is not optimized for low-latency queries. You need to use another service, such as Amazon Athena or Amazon Redshift Spectrum, to query the data in S3, which may incur additional costs and latency.
Option B is incorrect because it suggests using AWS Lambda to process the sensor data and store the data in Amazon S3 for querying. This solution has the following disadvantages:
? It does not allow you to ingest sensor data in near real time, as Lambda is a serverless compute service that runs code in response to events. You need to use another service, such as API Gateway or Kinesis Data Streams, to trigger Lambda functions with sensor data, which may add extra latency and complexity to your solution.
? It does not allow you to query the data with a latency of less than 10 milliseconds, as Amazon S3 is an object storage service that is not optimized for low-latency queries. You need to use another service, such as Amazon Athena or Amazon Redshift Spectrum, to query the data in S3, which may incur additional costs and latency.
Option D is incorrect because it suggests using Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data and use AWS Glue to store the data in Amazon RDS for querying. This solution has the following disadvantages:

? It does not allow you to ingest sensor data in near real time, as Amazon SQS is a message queue service that delivers messages in a best-effort manner. You need to use another service, such as Lambda or EC2, to poll the messages from the queue and process them, which may add extra latency and complexity to your solution.
? It does not allow you to store the data in nested JSON format, as Amazon RDS is a relational database service that supports structured data types, such as tables and columns. You need to use another service, such as AWS Glue, to transform the data from JSON to relational format, which may add extra cost and overhead to your solution.
References:
? 1: Amazon Kinesis Data Streams - Features
? 2: Amazon DynamoDB - Features
? 3: Loading Streaming Data into Amazon DynamoDB - Amazon Kinesis Data Firehose
? [4]: Capturing Table Activity with DynamoDB Streams - Amazon DynamoDB
? [5]: Amazon DynamoDB Accelerator (DAX) - Features
? [6]: Amazon S3 - Features
? [7]: AWS Lambda - Features
? [8]: Amazon Simple Queue Service - Features
? [9]: Amazon Relational Database Service - Features
? [10]: Working with JSON in Amazon RDS - Amazon Relational Database Service
? [11]: AWS Glue - Features

NEW QUESTION 9
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

Answer: C

Explanation:
Athena data source is a solution that allows you to use Spark to access Athena by using the Athena JDBC driver and the Spark SQL interface. You can use the Athena data source to create Spark DataFrames from Athena tables, run SQL queries on the DataFrames, and write the results back to Athena. The Athena data source supports various data formats, such as CSV, JSON, ORC, and Parquet, and also supports partitioned and bucketed tables. The Athena data source is a cost-effective and scalable way to use Spark to access Athena, as it does not require any additional infrastructure or services, and you only pay for the data scanned by Athena.
The other options are not solutions that give the company the ability to use Spark to access Athena. Option A, Athena query settings, is a feature that allows you to configure various parameters for your Athena queries, such as the output location, the encryption settings, the query timeout, and the workgroup. Option B, Athena workgroup, is a feature that allows you to isolate and manage your Athena queries and resources, such as the query history, the query notifications, the query concurrency, and the query cost. Option D, Athena query editor, is a feature that allows you to write and run SQL queries on Athena using the web console or the API. None of these options enable you to use Spark instead of SQL to generate analytics on Athena. References:
? Using Apache Spark in Amazon Athena
? Athena JDBC Driver
? Spark SQL
? Athena query settings
? [Athena workgroups]
? [Athena query editor]

NEW QUESTION 10
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 Catalo
B. Configure an AWS Glue Schema Registr
C. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.
D. Create an Amazon Redshift provisioned cluste
E. 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.
F. Create an Amazon Athena workgrou
G. Explore the data that is in Amazon S3 by using Apache Spark through Athen
H. Provide the Athena workgroup schema and tables to the analytics department.
I. Create an AWS Glue Data Catalo
J. Configure an AWS Glue Schema Registr
K. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data AP
L. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Answer: C

Explanation:
The best solution to meet the requirements of creating a data catalog that includes the IoT data, and allowing the analytics department to index the data, most cost- effectively, is to create an Amazon Athena workgroup, explore the data that is in Amazon S3 by using Apache Spark through Athena, and provide the Athena workgroup schema and tables to the analytics department.
Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python1. Amazon Athena also supports Apache Spark, an open-source distributed processing framework that can run large-scale data analytics applications across clusters of servers2. You can use Athena to run Spark code on data in Amazon S3 without having to set up, manage, or scale any infrastructure. You can also use Athena to create and manage external tables that point to your data in Amazon S3, and store them in an external data catalog, such as AWS Glue Data Catalog, Amazon Athena Data Catalog, or your own Apache Hive metastore3. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications4. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis5.

Using Athena and Spark to create a data catalog and explore the IoT data in Amazon S3 is the most cost-effective solution, as you pay only for the queries you run or the compute you use, and you pay nothing when the service is idle1. You also save on the operational overhead and complexity of managing data warehouse infrastructure, as Athena and Spark are serverless and scalable. You can also benefit from the flexibility and performance of Athena and Spark, as they support various data formats, including JSON, and can handle schema changes and complex queries efficiently.
Option A is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating a new AWS Glue workload to orchestrate theingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Glue Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components6. AWS Glue Schema Registry is a service that allows you to centrally store and manage the schemas of your streaming data in AWS Glue Data Catalog7. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores8. Amazon Redshift Serverless is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to run and scale analytics without having to manage data warehouse infrastructure9. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Glue charges you based on the compute time and the data processed by your ETL jobs8. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue and Amazon Redshift Serverless would introduce additional latency and complexity, as you would have to ingest the data from Amazon S3 to Amazon Redshift Serverless, and then query it from there, instead of querying it directly from Amazon S3 using Athena and Spark.
Option B is not the best solution, as creating an Amazon Redshift provisioned cluster, creating an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3, and creating Redshift stored procedures to load the data into Amazon Redshift, would incur more costs and complexity than using Athena and Spark. Amazon Redshift provisioned clusters are clusters that you create and manage by specifying the number and type of nodes, and the amount of storage and compute capacity10. Amazon Redshift Spectrum is a feature of Amazon Redshift that allows you to query and join data across your data warehouse and your data lake using standard SQL11. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command12. While these features are powerful and useful for many data warehousing scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. Amazon Redshift provisioned clusters charge you based on the node type, the number of nodes, and the duration of the cluster10. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries11. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using Amazon Redshift provisioned clusters and Spectrum would introduce additional latency and complexity, as you would have to provision andmanage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark. Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers13. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost- effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Lambda charges you based on the number of requests and the duration of your functions13. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark. References:
? What is Amazon Athena?
? Apache Spark on Amazon Athena
? Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs
? Managing Athena workgroups
? Using Amazon QuickSight to visualize data in Amazon Athena
? AWS Glue Data Catalog
? AWS Glue Schema Registry
? What is AWS Glue?
? Amazon Redshift Serverless
? Amazon Redshift provisioned clusters
? Querying external data using Amazon Redshift Spectrum
? Using stored procedures in Amazon Redshift
? What is AWS Lambda?
? [Creating and using AWS Lambda UDFs]
? [Using the Amazon Redshift Data API]
? [What is AWS Step Functions?]
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 10
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 storag
B. Use an Amazon Redshift provisioned cluster for data analysis.
C. Use Amazon S3 for data storag
D. Use Amazon Athena for data analysis.
E. Use AWS Glue DataBrewfor centralized data governance and access control.
F. Use Amazon RDS for data storag
G. Use Amazon EMR for data analysis.
H. Use AWS Lake Formation for centralized data governance and access control.

Answer: BE

Explanation:
A data mesh is an architectural framework that organizes data into domains and treats data as products that are owned and offered for consumption by different teams1. A data mesh requires a centralized layer for data governance and access control, as well as a distributed layer for data storage and analysis. AWS Glue can provide data catalogs and ETL operations for the data mesh, but it cannot provide data governance and access control by itself2. Therefore, the company needs to use another AWS service for this purpose. AWS Lake Formation is a service that allows you to create, secure, and manage data lakes on AWS3. It integrates with AWS Glue and other AWS services to provide centralized data governance and access control for the data mesh. Therefore, option E is correct.

For data storage and analysis, the company can choose from different AWS services depending on their needs and preferences. However, one of the benefits of a data mesh is that it enables data to be stored and processed in a decoupled and scalable way1. Therefore, using serverless or managed services that can handle large volumes and varieties of data is preferable. Amazon S3 is a highly scalable, durable, and secure object storage service that can store any type of data. Amazon Athena is a serverless interactive query service that can analyze data in Amazon S3 using standard SQL. Therefore, option B is a good choice for data storage and analysis in a data mesh. Option A, C, and D are not optimal because they either use relational databases that are not suitable for storing diverse and unstructured data, or they require more management and provisioning than serverless services. References:
? 1: What is a Data Mesh? - Data Mesh Architecture Explained - AWS
? 2: AWS Glue - Developer Guide
? 3: AWS Lake Formation - Features
? [4]: Design a data mesh architecture using AWS Lake Formation and AWS Glue
? [5]: Amazon S3 - Features
? [6]: Amazon Athena - Features

NEW QUESTION 12
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 applicatio
B. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.
C. Write a PySpark ETL scrip
D. Host the script on an Amazon EMR cluster.
E. Write an AWS Glue PySpark jo
F. Use Apache Spark to transform the data.
G. Write an AWS Glue Python shell jo
H. Use pandas to transform the data.

Answer: D

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle various data sources and formats, including .csv files in Amazon S3. AWS Glue provides two types of jobs: PySpark and Python shell. PySpark jobs use Apache Spark to process large-scale data in parallel, while Python shell jobs use Python scripts to process small-scale data in a single execution environment. For this requirement, a Python shell job is more suitable and cost-effective, as the size of each S3 object is less than 100 MB, which does not require distributed processing. A Python shell job can use pandas, a popular Python library fordata analysis, to transform the .csv data as needed. The other solutions are not optimal or relevant for this requirement. Writing a custom Python application and hosting it on an Amazon EKS cluster would require more effort and resources to set up and manage the Kubernetes environment, as well as to handle the data ingestion and transformation logic. Writing a PySpark ETL script and hosting it on an Amazon EMR cluster would also incur more costs and complexity to provision and configure the EMR cluster, as well as to use Apache Spark for processing small data files. Writing an AWS Glue PySpark job would also be less efficient and economical than a Python shell job, as it would involve unnecessary overhead and charges for using Apache Spark for small data files. References:
? AWS Glue
? Working with Python Shell Jobs
? pandas
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 17
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 EM
C. Migrate the existing on-premises Hive metastore into Amazon EM
D. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.
E. Configure an external Hive metastore in Amazon EM
F. Migrate the existing on-premises Hive metastore into Amazon EM
G. Use Amazon Aurora MySQL to store the company's data catalog.
H. Configure a new Hive metastore in Amazon EM
I. Migrate the existing on-premises Hive metastore into Amazon EM
J. Use the new metastore as the company's data catalog.

Answer: A

Explanation:
AWS Database Migration Service (AWS DMS) is a service that helps you migrate databases to AWS quickly and securely. You can use AWS DMS to migrate the Hive metastore from the on-premises Hadoop clusters into Amazon S3, which is a highlyscalable, durable, and cost-effective object storage service. AWS Glue Data Catalog is a serverless, managed service that acts as a central metadata repository for your data assets. You can use AWS Glue Data Catalog to scan the Amazon S3 bucket that contains the migrated Hive metastore and create a data catalog that is compatible with Apache Hive and other AWS services. This solution meets the requirements of migrating the data catalog into a persistent storage solution and using a serverless solution. This solution is also the most cost-effective, as it does not incur any additional charges for running Amazon EMR or Amazon Aurora MySQL clusters. The other options are either not feasible or not optimal. Configuring a Hive metastore in Amazon EMR (option B) or an external Hive metastore in Amazon EMR (option C) would require running and maintaining Amazon EMR clusters, which would incur additional costs and complexity. Using Amazon Aurora MySQL to store the company's data catalog (option C) would also incur additional costs and complexity, as well as introduce compatibility issues with Apache Hive. Configuring a new Hive metastore in Amazon EMR (option D) would not migrate the existing data catalog, but create a new one, which would result in data loss and inconsistency. References:
? Using AWS Database Migration Service
? Populating the AWS Glue Data Catalog
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.2: AWS Glue Data Catalog

NEW QUESTION 19
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

Answer: A

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle petabytes of data in seconds. AWS Glue can run Apache Spark and Apache Flink jobs without requiring any infrastructure provisioning or management. AWS Glue can also integrate with Apache Pig, Apache Oozie, and Apache Hbase using AWS Glue Data Catalog and AWS Glue workflows. AWS Glue can reduce the overall operational overhead by automating the data discovery, data preparation, and data loading processes. AWS Glue can also optimize the cost and performance of ETL jobs by using AWS Glue Job Bookmarking, AWS Glue Crawlers, and AWS Glue Schema Registry. References:
? AWS Glue
? AWS Glue Data Catalog
? AWS Glue Workflows
? [AWS Glue Job Bookmarking]
? [AWS Glue Crawlers]
? [AWS Glue Schema Registry]
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 22
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 .csvdata to JSON format by fetchingonly 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.

Answer: CE

Explanation:
Using an S3 bucket that is in the same AWS Region where the company runs Athena queries can improve query performance by reducing data transfer latency and costs. Preprocessing the .csv data to Apache Parquet format can also improve query performance by enabling columnar storage, compression, and partitioning, which can reduce the amount of data scanned and fetched by the query. These solutions can optimize the storage solution for the POC test without requiring much effort or changes to the existing data pipeline. The other solutions are not optimal or relevant for this requirement. Adding a randomized string to the beginning of the keys in Amazon S3 can improve the throughput across partitions, but it can also make the data harder to query and manage. Using an S3 bucket that is in the same account that uses Athena to query the data does not have any significant impact on query performance, as long as the proper permissions are granted. Preprocessing the .csv data to JSON format does not offer any benefits over the .csv format, as both are row-based and verbose formats that require more data scanning and fetching than columnar formats like Parquet. References:
? Best Practices When Using Athena with AWS Glue
? Optimizing Amazon S3 Performance
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 26
A manufacturing company collects sensor data from its factory floor to monitor and enhance operational efficiency. The company uses Amazon Kinesis Data Streams to publish the data that the sensors collect to a data stream. Then Amazon Kinesis Data Firehose writes the data to an Amazon S3 bucket.
The company needs to display a real-time view of operational efficiency on a large screen in the manufacturing facility.
Which solution will meet these requirements with the LOWEST latency?

A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor dat
B. Use a connector for Apache Flink to write data to an Amazon Timestream databas
C. Use the Timestream database as a source to create a Grafana dashboard.
D. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is create
E. Use the Lambda function to publish the data to Amazon Auror
F. Use Aurora as a source to create an Amazon QuickSight dashboard.
G. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor dat
H. Create a new Data Firehose delivery stream to publish data directly to an Amazon Timestream databas
I. Use the Timestream database as a source to create an Amazon QuickSight dashboard.
J. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real tim
K. Publish the data to an Amazon Timestream databas
L. Use the Timestream database as a source to create a Grafana dashboard.

Answer: C

Explanation:
This solution will meet the requirements with the lowest latency because it uses Amazon Managed Service for Apache Flink to process the sensor data in real time and write it to Amazon Timestream, a fast, scalable, and serverless time series database. Amazon Timestream is optimized for storing and analyzing time

series data, such as sensor data, and can handle trillions of events per day with millisecond latency. By using AmazonTimestream as a source, you can create an Amazon QuickSight dashboard that displays a real-time view of operational efficiency on a large screen in the manufacturing facility. Amazon QuickSight is a fully managed business intelligence service that can connect to various data sources, including Amazon Timestream, and provide interactive visualizations and insights123.
The other options are not optimal for the following reasons:
? A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is similar to option C, but it uses Grafana instead of Amazon QuickSight to create the dashboard. Grafana is an open source visualization tool that can also connect to Amazon Timestream, but it requires additional steps to set up and configure, such as deploying a Grafana server on Amazon EC2, installing the Amazon Timestream plugin, and creating an IAM role for Grafana to access Timestream. These steps can increase the latency and complexity of the solution.
? B. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard. This option is not suitable for displaying a real-time view of operational efficiency, as it introduces unnecessary delays and costs in the data pipeline. First, the sensor data is written to an S3 bucket by Amazon Kinesis Data Firehose, which can have a buffering interval of up to 900 seconds. Then, the S3 bucket sends a notification to a Lambda function, which can incur additional invocation and execution time. Finally, the Lambda function publishes the data to Amazon Aurora, a relational database that is not optimized for time series data and can have higher storage and performance costs than Amazon Timestream .
? D. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time.
Publish the data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is also not suitable for displaying a real-time view of operational efficiency, as it uses AWS Glue bookmarks to read sensor data from the S3 bucket. AWS Glue bookmarks are a feature that helps AWS Glue jobs and crawlers keep track of the data that has already been processed, so that they can resume from where they left off. However, AWS Glue jobs and crawlers are not designed for real-time data processing, as they can have a minimum frequency of 5 minutes and a variable start-up time. Moreover, this option also uses Grafana instead of Amazon QuickSight to create the dashboard, which can increase the latency and complexity of the solution .
References:
? 1: Amazon Managed Streaming for Apache Flink
? 2: Amazon Timestream
? 3: Amazon QuickSight
? : Analyze data in Amazon Timestream using Grafana
? : Amazon Kinesis Data Firehose
? : Amazon Aurora
? : AWS Glue Bookmarks
? : AWS Glue Job and Crawler Scheduling

NEW QUESTION 31
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 sourc
C. Configure the job to ingest the data into the data lake in JSON format.
D. Use an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format.
E. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data sourc
F. Configure the job to write the data into the data lake in Apache Parquet format.

Answer: D

Explanation:
Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports various data formats, such as CSV,JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row- oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.
On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance.
Therefore, creating an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source and writing the data into the data lake in Apache Parquet format will meet the requirements most cost-effectively. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue ETL jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API). By using AWS Glue ETL jobs, you can easily convert the data from CSV to Parquet format, without having to write or manage any code. Parquet is a column-oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3.
The other options are not as cost-effective as creating an AWS Glue ETL job to write the data into the data lake in Parquet format. Using an AWS Glue PySpark job to ingest the source data into the data lake in .csv format will not improve the query performance or reduce the query cost, as .csv is a row-oriented format that does not support columnar access or compression. Creating an AWS Glue ETL job to ingest the data into the data lake in JSON format will not improve the query performance or reduce the query cost, as JSON is also a row-oriented format that does not support columnar access or compression. Using an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format will improve the query performance, as Avro is a column-oriented format that supports compression and encoding, but it will require more operational effort, as you will need to write and maintain PySpark code to convert the data from CSV to Avro format. References:
? Amazon Athena
? Choosing the Right Data Format
? AWS Glue
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

NEW QUESTION 32
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

Answer: A

Explanation:
AWS Glue workflows are a feature of AWS Glue that enable you to create and visualize complex ETL pipelines using AWS Glue components, such as crawlers, jobs, triggers, anddevelopment endpoints. AWS Glue workflows provide automated orchestration and require minimal manual effort, as they handle dependency resolution, error handling, state management, and resource allocation for your ETL workflows. You can use AWS Glue workflows to ingest data from your operational databases into your Amazon S3 based data lake, and then use AWS Glue and Amazon EMR to process the data in the data
lake. This solution will meet the requirements with the least operational overhead, as it leverages the serverless and fully managed nature of AWS Glue, and the scalability and flexibility of Amazon EMR12.
The other options are not optimal for the following reasons:
? B. AWS Step Functions tasks. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. You can use AWS Step Functions tasks to invoke AWS Glue and Amazon EMR jobs as part of your ETL workflows, and use AWS Step Functions state machines to define the logic and flow of your workflows. However, this option would require more manual effort than AWS Glue workflows, as you would need to write JSON code to define your state machines, handle errors and retries, and monitor the execution history and status of your workflows3.
? C. AWS Lambda functions. AWS Lambda is a service that lets you run code without provisioning or managing servers. You can use AWS Lambda functions to trigger AWS Glue and Amazon EMR jobs as part of your ETL workflows, and use AWS Lambda event sources and destinations to orchestrate the flow of your workflows. However, this option would also require more manual effort than AWS Glue workflows, as you would need to write code to implement your business logic, handle errors and retries, and monitor the invocation and execution of your Lambda functions. Moreover, AWS Lambda functions have limitations on the execution time, memory, and concurrency, which may affect the performance and scalability of your ETL workflows.
? D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) workflows.
Amazon MWAA is a managed service that makes it easy to run open source Apache Airflow on AWS. Apache Airflow is a popular tool for creating and managing complex ETL pipelines using directed acyclic graphs (DAGs). You can use Amazon MWAA workflows to orchestrate AWS Glue and Amazon EMR jobs as part of your ETL workflows, and use the Airflow web interface to visualize and monitor your workflows. However, this option would have more operational overhead than AWS Glue workflows, as you would need to set up and configure your Amazon MWAA environment, write Python code to define your DAGs, and manage the dependencies and versions of your Airflow plugins and operators.
References:
? 1: AWS Glue Workflows
? 2: AWS Glue and Amazon EMR
? 3: AWS Step Functions
? : AWS Lambda
? : Amazon Managed Workflows for Apache Airflow

NEW QUESTION 37
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 Formatio
E. Add a data filter for each Region.
F. Create a separate S3 bucket for each Regio
G. Configure an IAM policy to allow S3 acces
H. Restrict access based on Region.

Answer: BD

Explanation:
AWS Lake Formation is a service that helps you build, secure, and manage data lakes on Amazon S3. You can use AWS Lake Formation to register the S3 path as a data lake location, and enable fine-grained access control to limit access to the records based on the HR department's Region. You can use data filters to specify which S3 prefixes or partitions each HR department can access, and grant permissions to the IAM roles of the HR departments accordingly. This solution will meet the requirement with the least operational overhead, as it simplifies the data lake management and security, and leverages the existing IAM roles of the HR departments12.
The other options are not optimal for the following reasons:
? A. Use data filters for each Region to register the S3 paths as data locations. This option is not possible, as data filters are not used to register S3 paths as data locations, but to grant permissions to access specific S3 prefixes or partitions within a data location. Moreover, this option does not specify how to limit access to the records based on the HR department's Region.
? C. Modify the IAM roles of the HR departments to add a data filter for each department's Region. This option is not possible, as data filters are not added to IAM roles, but to permissions granted by AWS Lake Formation. Moreover, this option does not specify how to register the S3 path as a data lake location, or how to enable fine-grained access control in AWS Lake Formation.
? E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region. This option is not recommended, as it would require more operational overhead to create and manage multiple S3 buckets, and to configure and maintain IAM policies for each HR department. Moreover, this option does not leverage the benefits of AWS Lake Formation, such as data cataloging, data transformation, and data governance.
References:
? 1: AWS Lake Formation
? 2: AWS Lake Formation Permissions
? : AWS Identity and Access Management
? : Amazon S3

NEW QUESTION 42
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

Answer: BC

Explanation:
Amazon Redshift Spectrum is a feature that allows you to run SQL queries directly against data in Amazon S3, without loading or transforming the data. Redshift Spectrum can query various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset ofcolumns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.
On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance.
Therefore, using a columnar storage file format, such as Parquet, will provide faster queries, as it allows Redshift Spectrum to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Additionally, partitioning the data based on the most common query predicates, such as date, time, region, etc., will provide faster queries, as it allows Redshift Spectrum to prune the partitions that do not match the query criteria, reducing the amount of data scanned from S3. Partitioning also improves the performance of joins and aggregations, as it reduces data skew and shuffling.
The other options are not as effective as using a columnar storage file format and partitioning the data. Using gzip compression to compress individual files to sizes that are between 1 GB and 5 GB will reduce the data size, but it will not improve the query performance significantly, as gzip is not a splittable compression algorithm and requires decompression before reading. Splitting the data into files that are less than 10 KB will increase the number of files and the metadata overhead, which will degrade the query performance. Using file formats that are not supported by Redshift Spectrum, such as XML, will not work, as Redshift Spectrum will not be able to read or parse the data. References:
? Amazon Redshift Spectrum
? Choosing the Right Data Format
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.3: Amazon Redshift Spectrum

NEW QUESTION 43
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 ha
B. Create multiple copies of the datase
C. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.
D. Create an S3 Object Lambda endpoin
E. Use the S3 Object Lambda endpoint to read data from the S3 bucke
F. Implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data.
G. Use AWS Glue to transform the data for each applicatio
H. Create multiple copies of the datase
I. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.
J. Create an API Gateway endpoint that has custom authorizer
K. Use the API Gateway endpoint to read data from the S3 bucke
L. Initiate a REST API call to dynamically redact PII based on the needs of each application that accesses the data.

Answer: B

Explanation:
Option B is the best solution to meet the requirements with the least operational overhead because S3 Object Lambda is a feature that allows you to add your own code to process data retrieved from S3 before returning it to an application. S3 Object Lambda works with S3 GET requests and can modify both the object metadata and the object data. By using S3 Object Lambda, you can implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data. This way, you can avoid creating and maintaining multiple copies of the dataset with different levels of redaction.
Option A is not a good solution because it involves creating and managing multiple copies of the dataset with different levels of redaction for each application. This option adds complexity and storage cost to the data protection process and requires additional resources and configuration. Moreover, S3 bucket policies cannot enforce fine-grained data access control at the row and column level, so they are not sufficient to redact PII.
Option C is not a good solution because it involves using AWS Glue to transform the data for each application. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. However, in this scenario, using AWS Glue to redact PII is not the best option because it requires creating and maintaining multiple copies of the dataset with different levels of redaction for each application. This option also adds extra time and cost to the data protection process and requires additional resources and configuration.
Option D is not a good solution because it involves creating and configuring an API Gateway endpoint that has custom authorizers. API Gateway is a service that allows youto create, publish, maintain, monitor, and secure APIs at any scale. API Gateway can also integrate with other AWS services, such as Lambda, to provide custom logic for processing requests. However, in this scenario, using API Gateway to redact PII is not the best option because it requires writing and maintaining custom code and configuration for the API endpoint, the custom authorizers, and the REST API call. This option also adds complexity and latency to the data protection process and requires additional resources and configuration.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? Introducing Amazon S3 Object Lambda – Use Your Code to Process Data as It Is Being Retrieved from S3
? Using Bucket Policies and User Policies - Amazon Simple Storage Service

? AWS Glue Documentation
? What is Amazon API Gateway? - Amazon API Gateway

NEW QUESTION 46
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. Confiqure 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 object
E. Use a SQL SELECT statement in Amazon Athena to query the required column.

Answer: B

Explanation:
Option B is the best solution to meet the requirements with the least operational overhead because S3 Select is a feature that allows you to retrieve only a subset of data from an S3 object by using simple SQL expressions. S3 Select works on objects stored in CSV, JSON, or Parquet format. By using S3 Select, you can avoid the need to download and process the entire S3 object, which reduces the amount of data transferred and the computation time. S3 Select is also easy to use and does not require any additional services or resources.
Option A is not a good solution because it involves writing custom code and configuring an AWS Lambda function to load data from the S3 bucket into a pandas dataframe and query the required column. This option adds complexity and latency to the data retrieval process and requires additional resources and configuration.Moreover, AWS Lambda has limitations on the execution time, memory, and concurrency, which may affect the performance and reliability of the data retrieval process.
Option C is not a good solution because it involves creating and running an AWS Glue DataBrew project to consume the S3 objects and query the required column. AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. However, in this scenario, the data is already in Parquet format, which is a columnar storage format that is optimized for analytics. Therefore, there is no need to use AWS Glue DataBrew to prepare the data. Moreover, AWS Glue DataBrew adds extra time and cost to the data retrieval process and requires additional resources and configuration.
Option D is not a good solution because it involves running an AWS Glue crawler on the S3 objects and using a SQL SELECT statement in Amazon Athena to query the required column. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. However, in this scenario, the schema and format of the data are already known and fixed, so there is no need to run a crawler to discover them. Moreover, running a crawler and using Amazon Athena adds extra time and cost to the data retrieval process and requires additional services and configuration.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? S3 Select and Glacier Select - Amazon Simple Storage Service
? AWS Lambda - FAQs
? What Is AWS Glue DataBrew? - AWS Glue DataBrew
? Populating the AWS Glue Data Catalog - AWS Glue
? What is Amazon Athena? - Amazon Athena

NEW QUESTION 50
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 functio
B. Program the Lambda function to write the event to Amazon Kinesis Data Firehos
C. Configure Kinesis Data Firehose to write the event to the logs S3 bucket.
D. Create a trail of management events in AWS CloudTrai
E. Configure the trail to receive data from the transactions S3 bucke
F. Specify an empty prefix and write-only event
G. Specify the logs S3 bucket as the destination bucket.
H. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda functio
I. Program the Lambda function to write the events to the logs S3 bucket.
J. Create a trail of data events in AWS CloudTrai
K. Configure the trail to receive data from the transactions S3 bucke
L. Specify an empty prefix and write-only event
M. Specify the logs S3 bucket as the destination bucket.

Answer: D

Explanation:
This solution meets the requirement of logging all writes to the S3 bucket into another S3 bucket with the least operational effort. AWS CloudTrail is a service that records the API calls made to AWS services, including Amazon S3. By creating a trail of data events, you can capture the details of the requests that are made to the transactions S3 bucket, such as the requester, the time, the IP address, and the response elements. By specifying an empty prefix and write-only events, you can filter the data events to only include the ones that write to the bucket. By specifying the logs S3 bucket as the destination bucket, you can store the CloudTrail logs in another S3 bucket that is in the same AWS Region. This solution does not require any additional coding or configuration, and it is more scalable and reliable than using S3 Event Notifications and Lambda functions. References:
? Logging Amazon S3 API calls using AWS CloudTrail
? Creating a trail for data events
? Enabling Amazon S3 server access logging

NEW QUESTION 51
A company wants to implement real-time analytics capabilities. The company wants to use Amazon Kinesis Data Streams and Amazon Redshift to ingest and process streaming data at the rate of several gigabytes per second. The company wants to derive near real-time insights by using existing business intelligence

(BI) and analytics tools.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use Kinesis Data Streams to stage data in Amazon S3. Use the COPY command to load data from Amazon S3 directly into Amazon Redshift to make the data immediately available for real-time analysis.
B. Access the data from Kinesis Data Streams by using SQL querie
C. Create materialized views directly on top of the strea
D. Refresh the materialized views regularly to query the most recent stream data.
E. Create an external schema in Amazon Redshift to map the data from Kinesis Data Streams to an Amazon Redshift objec
F. Create a materialized view to read data from the strea
G. Set the materialized view to auto refresh.
H. Connect Kinesis Data Streams to Amazon Kinesis Data Firehos
I. Use Kinesis Data Firehose to stage the data in Amazon S3. Use the COPY command to load the data from Amazon S3 to a table in Amazon Redshift.

Answer: C

Explanation:
This solution meets the requirements of implementing real-time analytics capabilities with the least operational overhead. By creating an external schema in Amazon Redshift, you can access the data from Kinesis Data Streams using SQL queries without having to load the data into the cluster. By creating a materialized view on top of the stream, you can store the results of the query in the cluster and make them available for analysis. By setting the materialized view to auto refresh, you can ensure that the view is updated with the latest data from the stream at regular intervals. This way, you can derive near real-time insights by using existing BI and analytics tools. References:
? Amazon Redshift streaming ingestion
? Creating an external schema for Amazon Kinesis Data Streams
? Creating a materialized view for Amazon Kinesis Data Streams

NEW QUESTION 52
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.

Answer: D

Explanation:
The error message indicates that the AWS Glue job cannot access the Amazon S3 bucket through the VPC endpoint. This could be because the VPC's route table does not have the necessary routes to direct the traffic to the endpoint. To fix this, the data engineer must verify that the route table has an entry for the Amazon S3 service prefix (com.amazonaws.region.s3) with the target as the VPC endpoint ID. This will allow the AWS Glue job to use the VPC endpoint to access the S3 bucket without going through the internet or a NAT gateway. For more information, see Gateway endpointsR. eferences:
? Troubleshoot the AWS Glue error "VPC S3 endpoint validation failed"
? Amazon VPC endpoints for Amazon S3
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 54
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 dat
B. Set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM role
C. Assign each user to the IAM role that matches the user's PII access requirements.
D. Use Amazon QuickSight to access the dat
E. Use column-level security features in QuickSight to limit the PII that users can retrieve from Amazon S3 by using Amazon Athen
F. Define QuickSight access levels based on the PII access requirements of the users.
G. Build a custom query builder UI that will run Athena queries in the background to access the dat
H. Create user groups in Amazon Cognit
I. Assign access levels to the user groups based on the PII access requirements of the users.
J. Create IAM roles that have different levels of granular acces
K. Assign the IAM roles to IAM user group
L. Use an identity-based policy to assign access levels to user groups at the column level.

Answer: A

Explanation:
Amazon Athena is a serverless, interactive query service that enables you to analyze data in Amazon S3 using standard SQL. AWS Lake Formation is a service that helps you build, secure, and manage data lakes on AWS. You can use AWS Lake Formation to create data filters that define the level of access for different IAM roles based on the columns, rows, or tags of the data. By using Amazon Athena to query the data and AWS Lake Formation to create data filters, the company can meet the requirements of ensuring that user groups can access only the PII that they require with the least effort. The solution is to use Amazon Athena to query the data in the data lake that is in Amazon S3. Then, set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. For example, a data filter can allow a user group to access only the columns that contain the PII that they need, such as name and email address, and deny access to the columns that contain the PII that they do not need, such as phone number and social security number. Finally, assign each user to the IAM role that matches the user's PII access requirements. This way, the user groups can access the data in the data lake securely and efficiently. The other options are either not feasible or not optimal. Using Amazon QuickSight to access the data (option B) would require the company to pay for the QuickSight service and to configure the column-level security features for each user. Building a custom query builder UI that will run Athena queries in the background to access the data (option C) would require the company to develop and maintain the UI and to integrate it with Amazon Cognito. Creating IAM roles that have different levels of granular access (option D) would require the company to manage multiple IAM roles and policies and to ensure that they are aligned with the data schema.

References:
? Amazon Athena
? AWS Lake Formation
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.3: Amazon Athena

NEW QUESTION 59
A data engineer needs Amazon Athena queries to finish faster. The data engineer notices that all the files the Athena queries use are currently stored in uncompressed .csv format. The data engineer also notices that users perform most queries by selecting a specific column.
Which solution will MOST speed up the Athena query performance?

A. Change the data format from .csvto JSON forma
B. Apply Snappy compression.
C. Compress the .csv files by using Snappy compression.
D. Change the data format from .csvto Apache Parque
E. Apply Snappy compression.
F. Compress the .csv files by using gzjg compression.

Answer: C

Explanation:
Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row- oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.
On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance.
Therefore, changing the data format from CSV to Parquet and applying Snappy compression will most speed up the Athena query performance. Parquet is a column- oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Snappy is a compression algorithm that reduces the data size without compromising the query speed, as it is splittable and does not require decompression before reading. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3.
The other options are not as effective as changing the data format to Parquet and applying Snappy compression. Changing the data format from CSV to JSON and applying Snappy compression will not improve the query performance significantly, as JSON is also a row- oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using Snappy compression will reduce the data size, but it will not improve the query performance significantly, as CSV is still a row-oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using gzjg compression will reduce the data size, but it willdegrade the query performance, as gzjg is not a splittable compression algorithm and requires decompression before reading. References:
? Amazon Athena
? Choosing the Right Data Format
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

NEW QUESTION 60
......