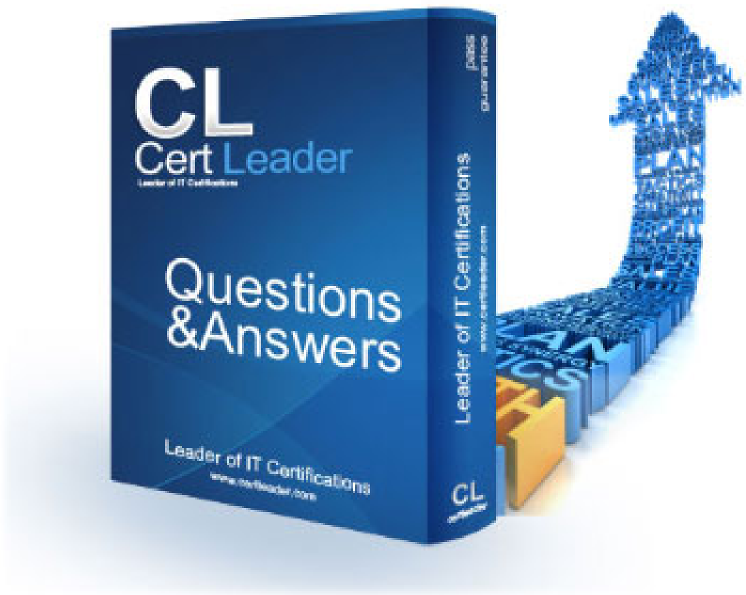
AWS-Certified-Data-Engineer-Associate Dumps

AWS Certified Data Engineer - Associate (DEA-C01)

https://www.certleader.com/AWS-Certified-Data-Engineer-Associate-dumps.html



NEW QUESTION 1
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 2
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 asa consumer of the ETL cluster by using Redshift data sharing.
B. Create materialized views based on the sales team's requirement
C. Grant the sales team direct access to the ETL cluster.
D. Create database views based on the sales team's requirement
E. Grant the sales team direct access to the ETL cluster.
F. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every wee
G. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

Answer: A

Explanation:
Redshift data sharing is a feature that enables you to share live data across different Redshift clusters without the need to copy or move data. Data sharing provides secure and governed access to data, while preserving the performance and concurrency benefits of Redshift. By setting up the sales team BI cluster as a consumer of the ETL cluster, the company can share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution also minimizes the usage of the computing resources of the ETL cluster, as the data sharing does not consume any storage space or compute resources from the producer cluster. The other options are either not feasible or not efficient. Creating materialized views or database views would require the sales team to have direct access to the ETL cluster, which could interfere with the critical analysis tasks. Unloading a copy of the data from the ETL cluster to anAmazon S3 bucket every week would introduce additional latency and cost, as well as create data inconsistency issues. References:
? Sharing data across Amazon Redshift clusters
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.2: Amazon Redshift

NEW QUESTION 3
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 deploymen
B. Create a step to pause the state machine during the EMR jobs that fai
C. Configure the step to wait for a human user to send approval through an email messag
D. Include details of the EMR task in the email message for further analysis.
E. Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR job
F. Verify that the Step Functions state machine code also includes IAM permissions to access the Amazon S3 buckets that the EMR jobs us
G. Use Access Analyzer for S3 to check the S3 access properties.
H. Check for entries in Amazon CloudWatch for the newly created EMR cluste
I. Change the AWS Step Functions state machine code to use Amazon EMR on EK
J. 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).
K. Query the flow logs for the VP
L. Determine whether the traffic that originates from the EMR cluster can successfully reach the data provider
M. 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.
N. Check the retry scenarios that the company configured for the EMR job
O. Increase the number of seconds in the interval between each EMR tas
P. Validate that each fallback state has the appropriate catch for each decision stat
Q. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

Answer: BD

Explanation:
To identify the reason why the Step Functions state machine is not able to run the EMR jobs, the company should take the following steps:
? Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. The state machine code should have an IAM role that allows it to invoke the EMR APIs, such as RunJobFlow, AddJobFlowSteps, and DescribeStep. The state machine code should also have IAM permissions to access the Amazon S3 buckets that the EMR jobs use as input and output locations. The company can use Access Analyzer for S3 to check the access policies and permissions of the S3 buckets12. Therefore, option B is correct.
? Query the flow logs for the VPC. The flow logs can provide information about the network traffic to and from the EMR cluster that is launched in the VPC. The company can use the flow logs to determine whether the traffic that originates from the EMR cluster can successfully reach the data providers, such as Amazon RDS, Amazon Redshift, or other external sources. The company can also determine whether any security group that might be attached to the EMR cluster allows connections to the data source servers on the informed ports. The company can use Amazon VPC Flow Logs or Amazon CloudWatch Logs Insights to query the flow logs3 . Therefore, option D is correct.
Option A is incorrect because it suggests using AWS CloudFormation to automate the Step Functions state machine deployment. While this is a good practice to ensure consistency and repeatability of the deployment, it does not help to identify the reasonwhy the state machine is not able to run the EMR jobs. Moreover, creating a step to pause the state machine during the EMR jobs that fail and wait for a human user to send approval through an email message is not a reliable way to troubleshoot the issue. The company should use the Step Functions console or API to monitor the execution history and status of the state machine, and use Amazon CloudWatch to view the logs and metrics of the EMR jobs . Option C is incorrect because it suggests changing the AWS Step Functions state machine code to use Amazon EMR on EKS. Amazon EMR on EKS is a service that allows you to run EMR jobs on Amazon Elastic Kubernetes Service (Amazon EKS) clusters. While this service has some benefits, such as lower cost and faster execution time, it does not support all the features and integrations that EMR on EC2 does, such as EMR Notebooks, EMR Studio, and EMRFS. Therefore, changing the state machine code to use EMR on EKS may not be compatible with the existing data pipeline and may introduce new issues. Option E is incorrect because it suggests checking the retry scenarios that the company configured for the EMR jobs. While this is a good practice to handle transient failures and errors, it does not help to identify the root cause of why the state machine is not able to run the EMR jobs. Moreover, increasing the number of seconds in the interval between each EMR task may not improve the success rate of the jobs, and may increase the execution time and cost of the state machine. Configuring an Amazon SNS topic to store the error messages may help to notify the company of any failures, but it does not provide enough information to troubleshoot the issue.
References:
? 1: Manage an Amazon EMR Job - AWS Step Functions
? 2: Access Analyzer for S3 - Amazon Simple Storage Service
? 3: Working with Amazon EMR and VPC Flow Logs - Amazon EMR
? [4]: Analyzing VPC Flow Logs with Amazon CloudWatch Logs Insights - Amazon Virtual Private Cloud
? [5]: Monitor AWS Step Functions - AWS Step Functions
? [6]: Monitor Amazon EMR clusters - Amazon EMR
? [7]: Amazon EMR on Amazon EKS - Amazon EMR

NEW QUESTION 4
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 table
B. Specify primary and foreign keys for all tables.
C. Use the ALL distribution style for large table
D. Specify primary and foreign keys for all tables.
E. Use the ALL distribution style for rarely updated small table
F. Specify primary and foreign keys for all tables.
G. Specify a combination of distribution, sort, and partition keys for all tables.

Answer: C

Explanation:
This solution meets the requirements of optimizing the performance of table SQL queries without increasing the size of the cluster. By using the ALL distribution style for rarely updated small tables, you can ensure that the entire table is copied to every node in the cluster, which eliminates the need for data redistribution during joins. This can improve query performance significantly, especially for frequently joined dimension tables. However, using the ALL distribution style also increases the storage space and the load time, so it is only suitable for small tables that are not updated frequently orextensively. By specifying primary and foreign keys for all tables, you can help the query optimizer to generate better query plans and avoid unnecessary scans or joins. You can also use the AUTO distribution style to let Amazon Redshift choose the optimal distribution style based on the table size and the query patterns. References:
? Choose the best distribution style
? Distribution styles
? Working with data distribution styles

NEW QUESTION 5
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 volume
B. Create new gp3 volumes from the snapshot
C. Attach the new gp3 volumes to the EC2 instances.
D. Create new gp3 volume
E. Gradually transfer the data to the new gp3 volume
F. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.
G. Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.
H. Use AWS DataSync to create new gp3 volume
I. Transfer the data from the original gp2 volumes to the new gp3 volumes.

Answer: C

Explanation:

Changing the volume type of the existing gp2 volumes to gp3 is the easiest and fastest way to migrate to the new storage type without any downtime or data loss. You can use the AWS Management Console, the AWS CLI, or the Amazon EC2 API to modify the volume type, size, IOPS, and throughput of your gp2 volumes. The modification takes effect immediately, and you can monitor the progress of the modification using CloudWatch. The other options are either more complex or require additional steps, such as creating snapshots, transferring data, or attaching new volumes, which can increase the operational overhead and the risk of errors. References:
? Migrating Amazon EBS volumes from gp2 to gp3 and save up to 20% on
costs (Section: How to migrate from gp2 to gp3)
? Switching from gp2 Volumes to gp3 Volumes to Lower AWS EBS Costs (Section: How to Switch from GP2 Volumes to GP3 Volumes)
? Modifying the volume type, IOPS, or size of an EBS volume - Amazon Elastic Compute Cloud (Section: Modifying the volume type)

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. QuickSiqht 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 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 catalo
B. Create AWS Lambda functions that will connect to the data catalo
C. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalo
D. Schedule the Lambda functions to run periodically.
E. Use the AWS Glue Data Catalog as the central metadata repositor
F. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata change
G. Schedule the crawlers to run periodically to update the metadata catalog.
H. Use Amazon DynamoDB as the data catalo
I. Create AWS Lambda functions that will connect to the data catalo
J. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalo
K. Schedule the Lambda functions to run periodically.
L. Use the AWS Glue Data Catalog as the central metadata repositor
M. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalo
N. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

Answer: B

Explanation:
This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions12.
The other options are not optimal for the following reasons:
? 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. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.
? 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. This option is also not recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.
? 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. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog.
References:
? 1: AWS Glue Data Catalog
? 2: AWS Glue Crawlers
? : Amazon Aurora
? : AWS Lambda
? : Amazon DynamoDB

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 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 joinsacross 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 element
B. Create an AWS Glue job that selects the data directly from the view and transfers the data in Parquet format to an S3 bucke
C. Schedule the AWS Glue job to run every day.
D. Schedule SQL Server Agent to run a daily SQL query that selects the desired data elements from the EC2 instance-based SQL Server database
E. Configure the query to direct the output .csv objects to an S3 bucke
F. Create an S3 event that invokes an AWS Lambda function to transform the output format from .csv to Parquet.
G. Use a SQL query to create a view in the EC2 instance-based SQL Server databases that contains the required data element
H. Create and run an AWS Glue crawler to read the vie
I. Create an AWS Glue job that retrieves the data and transfers the data in Parquet format to an S3 bucke
J. Schedule the AWS Glue job to run every day.
K. 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 bucke
L. Use Amazon EventBridge to schedule the Lambda function to run every day.

Answer: A

Explanation:
Option A is the most operationally efficient way to meet the requirements because it minimizes the number of steps and services involved in the data export process. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including Amazon S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. By creating a view in the SQL Server databases that contains the required data elements, the AWS Glue job can select the data directly from the view without having to perform any joins or transformations on the source data. The AWS Glue job can then transfer the data in Parquet format to an S3 bucket and run on a daily schedule.
Option B is not operationally efficient because it involves multiple steps and services to export the data. SQL Server Agent is a tool that can run scheduled tasks on SQL Server databases, such as executing SQL queries. However, SQL Server Agent cannot directly export data to S3, so the query output must be saved as .csv objects on the EC2 instance. Then, an S3 event must be configured to trigger an AWS Lambda function that can transform the .csv objects to Parquet format and upload them to S3. This option adds complexity and latency to the data export process and requires additional resources and configuration.
Option C is not operationally efficient because it introduces an unnecessary step of running an AWS Glue crawler to read the view. 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. However, in this scenario, the schema and format of the data elements are already known and fixed, so there is no need to run a crawler to discover them. The AWS Glue job can directly select the data from the view without using the Data Catalog. Running a crawler adds extra time and cost to the data export process.
Option D is not operationally efficient because it requires custom code and configuration to query the databases and transform the data. An AWS Lambda function is a service that can run code in response to events or triggers, such as Amazon EventBridge. Amazon EventBridge is a service that can connect applications and services with event sources, such as schedules, and route them to targets, such as Lambda functions. However, in this scenario, using a Lambda function to query the databases and transform the data is not the best option because it requires writing and maintaining code that uses JDBC to connect to the SQL Server

databases, retrieve the required data, convert the data to Parquet format, and transfer the data to S3. This option also has limitations on the execution time, memory, and concurrency of the Lambda function, which may affect the performance and reliability of the data export process.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? AWS Glue Documentation
? Working with Views in AWS Glue
? Converting to Columnar Formats

NEW QUESTION 14
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 utilizatio
B. Optimize the problematic queries.
C. Modify the database schema to include additional tables and indexes.
D. Reboot the RDS DB instance once each week.
E. Upgrade to a larger instance size.
F. Implement caching to reduce the database query load.

Answer: AE

Explanation:
Amazon RDS is a fully managed service that provides relational databases in the cloud. Amazon RDS for MySQL is one of the supported database engines that you can use to run your applications. Amazon RDS provides various features and tools to monitor and optimize the performance of your DB instances, such as Performance Insights, Enhanced Monitoring, CloudWatch metrics and alarms, etc.
Using the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization and optimizing the problematic queries will help reduce the CPU utilization of the DB instance. Performance Insights is a feature that allows you to analyze the load on your DB instance and determine what is causing performance issues. Performance Insights collects, analyzes, and displays database performance data using an interactive dashboard. You can use Performance Insights to identify the top SQL statements, hosts, users, or processes that are consuming the most CPU resources. You can also drill down into the details of each query and see the execution plan, wait events, locks, etc. By using Performance Insights, you can pinpoint the root cause of the high CPU utilization and optimize the queries accordingly. For example, you can rewrite the queries to make them more efficient, add or remove indexes, use prepared statements, etc.
Implementing caching to reduce the database query load will also help reduce the CPU utilization of the DB instance. Caching is a technique that allows you to store frequently accessed data in a fast and scalable storage layer, such as Amazon ElastiCache. By using caching, you can reduce the number of requests that hit your database, which in turn reduces the CPU load on your DB instance. Caching also improves the performance and availability of your application, as it reduces the latency and increases the throughput of your data access. You can use caching for various scenarios, such as storing session data, user preferences, application configuration, etc. You can also use caching for read-heavy workloads, such as displaying product details, recommendations, reviews, etc.
The other options are not as effective as using Performance Insights and caching. Modifying the database schema to include additional tables and indexes may or may not improve the CPU utilization, depending on the nature of the workload and the queries. Adding more tables and indexes may increase the complexity and overhead of the database, which may negatively affect the performance. Rebooting the RDS DB instance once each week will not reduce the CPU utilization, as it will not address the underlying cause of the high CPU load. Rebooting may also cause downtime and disruption to your application. Upgrading to a larger instance size may reduce the CPUutilization, but it will also increase the cost and complexity of your solution. Upgrading may also not be necessary if you can optimize the queries and reduce the database load by using caching. References:
? Amazon RDS
? Performance Insights
? Amazon ElastiCache
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 3: Data Storage and Management, Section 3.1: Amazon RDS

NEW QUESTION 16
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.

Answer: B

Explanation:
The Amazon Redshift Data API is a built-in feature that allows you to run SQL queries on Amazon Redshift data with web services-based applications, such as AWS Lambda, Amazon SageMaker notebooks, and AWS Cloud9. The Data API does not require a persistent connection to your database, and it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections. The Data API also supports both Amazon Redshift provisioned clusters and Redshift Serverless workgroups. The Data API is the best solution for running real-time queries on the financial data from within the trading application, as it has the least operational overhead compared to the other options.
Option A is not the best solution, as establishing WebSocket connections to Amazon Redshift would require more configuration and maintenance than using the Data API. WebSocket connections are also not supported by Amazon Redshift clusters or serverless workgroups.
Option C is not the best solution, as setting up JDBC connections to Amazon Redshift would also require more configuration and maintenance than using the Data API. JDBC connections are also not supported by Redshift Serverless workgroups.
Option D is not the best solution, as storing frequently accessed data in Amazon S3 and using Amazon S3 Select to run the queries would introduce additional latency and complexity than using the Data API. Amazon S3 Select is also not optimized for real-time queries, as it scans the entire object before returning the results. References:
? Using the Amazon Redshift Data API
? Calling the Data API
? Amazon Redshift Data API Reference
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 18

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

Answer: A

Explanation:
Option A is the correct answer because it meets the requirements with the least operational overhead. Creating an S3 event notification that has an event type of s3:ObjectCreated:* will trigger the Lambda function whenever a new object is created in the S3 bucket. Using a filter rule to generate notifications only when the suffix includes .csv will ensure that the Lambda function only runs for .csv files. Setting the ARN of the Lambda function as the destination for the event notification will directly invoke the Lambda function without any additional steps.
Option B is incorrect because it requires the user to tag the objects with .csv, which adds an extra step and increases the operational overhead.
Option C is incorrect because it uses an event type of s3:*, which will trigger the Lambda function for any S3 event, not just object creation. This could result in unnecessary invocations and increased costs.
Option D is incorrect because it involves creating and subscribing to an SNS topic, which adds an extra layer of complexity and operational overhead.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.2: S3 Event Notifications and Lambda Functions, Pages 67-69
? Building Batch Data Analytics Solutions on AWS, Module 4: Data Transformation, Lesson 4.2: AWS Lambda, Pages 4-8
? AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon S3, Pages 1-5

NEW QUESTION 19
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

Answer: C

Explanation:
Amazon Redshift is a fully managed, petabyte-scale data warehouse service that enables fast and cost-effective analysis of large volumes of data. Amazon Redshift uses columnar storage, compression, and zone maps to optimize the storage and performance of data. However, over time, as data is inserted, updated, or deleted, the physical storage of data can become fragmented, resulting in wasted disk space and degraded query performance. To address this issue, Amazon Redshift provides the VACUUM command, which reclaims disk space and resorts rows in either a specified table or all tables in the current schema1.
The VACUUM command has four options: FULL, DELETE ONLY, SORT ONLY, and REINDEX. The option that best meets the requirements of the question is VACUUM REINDEX, which re-sorts the rows in a table that has an interleaved sort key and rewritesthe table to a new location on disk. An interleaved sort key is a type of sort key that gives equal weight to each column in the sort key, and stores the rows in a way that optimizes the performance of queries that filter by multiple columns in the sort key. However, as data is added or changed, the interleaved sort order can become skewed, resulting in suboptimal query performance. The VACUUM REINDEX option restores the optimal interleaved sort order and reclaims disk space by removing deleted rows. This option also analyzes the sort key column and updates the table statistics, which are used by the query optimizer to generate the most efficient query execution plan23.
The other options are not optimal for the following reasons:
? A. VACUUM FULL Orders. This option reclaims disk space by removing deleted rows and resorts the entire table. However, this option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option is the most resource-intensive and time-consuming, as it rewrites the entire table to a new location on disk.
? B. VACUUM DELETE ONLY Orders. This option reclaims disk space by removing deleted rows, but does not resort the table. This option is not suitable for tables that have any sort key, as it does not improve the query performance by restoring the sort order. Moreover, this option does not analyze the sort key column and update the table statistics.
? D. VACUUM SORT ONLY Orders. This option resorts the entire table, but does not reclaim disk space by removing deleted rows. This option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option does not analyze the sort key column and update the table statistics.
References:
? 1: Amazon Redshift VACUUM
? 2: Amazon Redshift Interleaved Sorting
? 3: Amazon Redshift ANALYZE

NEW QUESTION 21
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 duringthe creation of andnew Redshift cluster.
D. Turn on concurrency scaling for the daily usage quota for the Redshift cluster.

Answer: B

Explanation:
Concurrency scaling is a feature that allows you to support thousands of concurrent users and queries, with consistently fast query performance. When you turn on concurrency scaling, Amazon Redshift automatically adds query processing power in seconds to process queries without any delays. You can manage which queries are sent to the concurrency-scaling cluster by configuring WLM queues. To turn on concurrency scaling for a queue, set the Concurrency Scaling mode value to auto. The other options are either incorrect or irrelevant, as they do not enable concurrency scaling for the existing Redshift cluster on RA3 nodes.
References:
? Working with concurrency scaling - Amazon Redshift
? Amazon Redshift Concurrency Scaling - Amazon Web Services
? Configuring concurrency scaling queues - Amazon Redshift
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 6, page 163)

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 29
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 polic
B. Associate the role with the crawle
C. Specify the S3 bucket path of the source data as the crawler's data stor
D. Create a daily schedule to run the crawle
E. Configure the output destination to a new path in the existing S3 bucket.
F. Create an IAM role that includes the AWSGlueServiceRole polic
G. Associate the role with the crawle

H. Specify the S3 bucket path of the source data as the crawler's data stor
I. Create a daily schedule to run the crawle
J. Specify a database name for the output.
K. Create an IAM role that includes the AmazonS3FullAccess polic
L. Associate the role with the crawle
M. Specify the S3 bucket path of the source data as the crawler's data stor
N. Allocate data processing units (DPUs) to run the crawler every da
O. Specify a database name for the output.
P. Create an IAM role that includes the AWSGlueServiceRole polic
Q. Associate the role with the crawle
R. Specify the S3 bucket path of the source data as the crawler's data stor
S. Allocate data processing units (DPUs) to run the crawler every da
T. Configure the output destination to a new path in the existing S3 bucket.

Answer: B

Explanation:
To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:
? Create an IAM role that has the necessary permissions to access the S3 data and
the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions1.
? Associate the role with the crawler.
? Specify the S3 bucket path of the source data as the crawler’s data store. The crawler will scan the data and infer the schema and format2.
? Create a daily schedule to run the crawler. The crawler will run at the specified time every day and update the Data Catalog with any changes in the data3.
? Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in the S3 bucket, such as the location, schema, and classification.
Option B is the only solution that includes all these steps. Therefore, option B is the correct answer.
Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data Catalog.
Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.
Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.
References:
? 1: AWS managed (predefined) policies for AWS Glue - AWS Glue
? 2: Data Catalog and crawlers in AWS Glue - AWS Glue
? 3: Scheduling an AWS Glue crawler - AWS Glue
? [4]: Parameters set on Data Catalog tables by crawler - AWS Glue
? [5]: AWS Glue pricing - Amazon Web Services (AWS)

NEW QUESTION 32
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 ra3.4xlarqe to ra3.16xlarqe.
D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Answer: B

Explanation:
Changing the distribution key to the table column that has the largest dimension will help to balance the load more evenly across all five compute nodes. The distribution key determines how the rows of a table are distributed among the slices of the cluster. If the distribution key is not chosen wisely, it can cause data skew, meaning some slices will have more data than others, resulting in uneven CPU load and query performance. By choosing the table column that has the largest dimension, meaning the column that has the most distinct values, as the distribution key, the data engineer can ensure that the rows are distributed more uniformly across the slices, reducing data skew and improving query performance.
The other options are not solutions that will meet the requirements. Option A, changing the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load. The sort key determines the order in which the rows of a table are stored on disk, which can improve the performance of range-restricted queries, but not the load balancing. Option C, upgrading the reserved node from ra3.4xlarge to ra3.16xlarge, will not maintain the current number of compute nodes, as it will increase the cost and the capacity of the cluster. Option D, changing the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load either. The primary key is a constraint that enforces the uniqueness of the rows in a table, but it does not influence the data layout or the query optimization. References:
? Choosing a data distribution style
? Choosing a data sort key
? Working with primary keys

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 39
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 41
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 44
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.

Answer: BD

Explanation:
The best combination of resources to meet the requirements of high reliability, cost-optimization, and performance for running Apache Spark jobs on Amazon EMR is to use Amazon S3 as a persistent data store and Graviton instances for core nodes and task nodes.
Amazon S3 is a highly durable, scalable, and secure object storage service that can store any amount of data for a variety of use cases, including big data analytics1. Amazon S3 is a better choice than HDFS as a persistent data store for Amazon EMR, as it decouples the storage from the compute layer, allowing for more flexibility and cost-efficiency. Amazon S3 also supports data encryption, versioning, lifecycle management, and cross-region replication1. Amazon EMR integrates seamlessly with Amazon S3, using EMR File System (EMRFS) to access data stored in Amazon S3 buckets2. EMRFS also supports consistent view, which enables Amazon EMR to provide read-after-write consistency for Amazon S3 objects that are accessed through EMRFS2.
Graviton instances are powered by Arm-based AWS Graviton2 processors that deliver up to 40% better price performance over comparable current generation x86-based instances3. Graviton instances are ideal for running workloads that are CPU-bound, memory-bound, or network-bound, such as big data analytics, web servers, and open- source databases3. Graviton instances are compatible with Amazon EMR, and can beused for both core nodes and task nodes. Core nodes are responsible for running the data processing frameworks, such as Apache Spark, and storing data in HDFS or the local file system. Task nodes are optional nodes that can be added to a cluster to increase the processing power and throughput. By using Graviton instances for both core nodes and task nodes, you can achieve higher performance and lower cost than using x86-based instances.
Using Spot Instances for all primary nodes is not a good option, as it can compromise the reliability and availability of the cluster. Spot Instances are spare EC2 instances that are available at up to 90% discount compared to On-Demand prices, but they can be interrupted by EC2 with a two-minute notice when EC2 needs the capacity back. Primary nodes are the nodes that run the cluster software, such as Hadoop, Spark, Hive, and Hue, and are essential for the cluster operation. If a primary node is interrupted by EC2, the cluster will fail or become unstable. Therefore, it is recommended to use On-Demand Instances or Reserved Instances for primary nodes, and use Spot Instances only for task nodes that can tolerate interruptions. References:
? Amazon S3 - Cloud Object Storage
? EMR File System (EMRFS)
? AWS Graviton2 Processor-Powered Amazon EC2 Instances
? [Plan and Configure EC2 Instances]
? [Amazon EC2 Spot Instances]
? [Best Practices for Amazon EMR]

NEW QUESTION 49
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.

Answer: C

Explanation:
The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:
? Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying
AWS Glue Data Catalog as the metastore)
? Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)
? AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

NEW QUESTION 52
A data engineer needs to use AWS Step Functions to design an orchestration workflow. The workflow must parallel process a large collection of data files and apply a specific transformation to each file.
Which Step Functions state should the data engineer use to meet these requirements?

A. Parallel state
B. Choice state
C. Map state
D. Wait state

Answer: C

Explanation:
Option C is the correct answer because the Map state is designed to process a collection of data in parallel by applying the same transformation to each element. The Map state can invoke a nested workflow for each element, which can be another state machine ora Lambda function. The Map state will wait until all the parallel executions are completed before moving to the next state.
Option A is incorrect because the Parallel state is used to execute multiple branches of logic concurrently, not to process a collection of data. The Parallel state can have different branches with different logic and states, whereas the Map state has only one branch that is applied to each element of the collection.
Option B is incorrect because the Choice state is used to make decisions based on a comparison of a value to a set of rules. The Choice state does not process any data or invoke any nested workflows.
Option D is incorrect because the Wait state is used to delay the state machine from continuing for a specified time. The Wait state does not process any data or invoke any nested workflows.
References:
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.3: AWS Step Functions, Pages 131-132
? Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.2: AWS Step Functions, Pages 9-10
? AWS Documentation Overview, AWS Step Functions Developer Guide, Step Functions Concepts, State Types, Map State, Pages 1-3

NEW QUESTION 53
......