Amazon

Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)

About Exambible

Your Partner of IT Exam

Found in 1998

Exambible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, Exambible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

NEW QUESTION 1
A data engineer needs to join data from multiple sources to perform a one-time analysis job. The data is stored in Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3.
Which solution will meet this requirement MOST cost-effectively?

A. Use an Amazon EMR provisioned cluster to read from all source
B. Use Apache Spark to join the data and perform the analysis.
C. Copy the data from DynamoDB, Amazon RDS, and Amazon Redshift into Amazon S3. Run Amazon Athena queries directly on the S3 files.
D. Use Amazon Athena Federated Query to join the data from all data sources.
E. Use Redshift Spectrum to query data from DynamoDB, Amazon RDS, and Amazon S3 directly from Redshift.

Answer: C

Explanation:
Amazon Athena Federated Query is a feature that allows you to query data from multiple sources using standard SQL. You can use Athena Federated Query to join data from Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3, as well as other data sources such as MongoDB, Apache HBase, and Apache Kafka1. Athena Federated Query is a serverless and interactive service, meaning you do not need to provision or manage any infrastructure, and you only pay for the amount of data scanned by your queries. Athena Federated Query is the most cost-effective solution for performing a one-time analysis job on data from multiple sources, as it eliminates the need to copy or move data, and allows you to query data directly from the source.
The other options are not as cost-effective as Athena Federated Query, as they involve additional steps or costs. Option A requires you to provision and pay for an Amazon EMR cluster, which can be expensive and time-consuming for a one-time job. Option B requires you to copy or move data from DynamoDB, RDS, and Redshift to S3, which can incur additional costs for data transfer and storage, and also introduce latency and complexity. Option D requires you to have an existing Redshift cluster, which can be costly and may not be necessary for a one-time job. Option D also does not supportquerying data from RDS directly, so you would need to use Redshift Federated Query to access RDS data, which adds another layer of complexity2. References:
? Amazon Athena Federated Query
? Redshift Spectrum vs Federated Query

NEW QUESTION 2
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 3
A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisionedcapacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use AWS Glue to crawl the data source
B. Store metadata in the AWS Glue Data Catalo
C. Use Amazon Athena to query the dat
D. Use SQL for structured data source
E. Use PartiQL for data that is stored in JSON format.
F. Use AWS Glue to crawl the data source
G. Store metadata in the AWS Glue Data Catalo
H. Use Redshift Spectrum to query the dat
I. Use SQL for structured data source
J. Use PartiQL for data that is stored in JSON format.
K. Use AWS Glue to crawl the data source
L. Store metadata in the AWS Glue Data Catalo
M. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv forma
N. Store the transformed data in an S3 bucke
O. Use Amazon Athena to query the original and transformed data from the S3 bucket.
P. Use AWS Lake Formation to create a data lak
Q. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet forma
R. Store the transformed data in an S3 bucke
S. Use Amazon Athena or Redshift Spectrum to query the data.

Answer: A

Explanation:
The best solution to meet the requirements of giving data scientists the ability to query all data sources by using syntax similar to SQL with the least operational

overhead is to use AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use Amazon Athena to query the data, use SQL for structured data sources, and use PartiQL for data that is stored in JSON format.
AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores1. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components3. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog.
Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python4. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries.
Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs1. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3.
Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL. Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster5. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena.
Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes6. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency, as you would have to wait for the ETL jobs to finish before querying the transformed data.
Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use Amazon Athena or RedshiftSpectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning7. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option B. References:
? What is Amazon Athena?
? Data Catalog and crawlers in AWS Glue
? AWS Glue Data Catalog
? Columnar Storage Formats
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
? 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?]
? [PartiQL for Amazon Athena]
? [Federated queries in Amazon Athena]
? [Amazon Athena pricing]
? [Top 10 performance tuning tips for Amazon Athena]
? [AWS Glue ETL jobs]
? [AWS Lake Formation jobs]

NEW QUESTION 4
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 5
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 6
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 7
A company uses Amazon S3 to store semi-structured data in a transactional data lake. Some of the data files are small, but other data files are tens of terabytes. A data engineer must perform a change data capture (CDC) operation to identify changed data from the data source. The data source sends a full snapshot as a JSON file every day and ingests the changed data into the data lake.
Which solution will capture the changed data MOST cost-effectively?

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

Answer: C

Explanation:
An open source data lake format, such as Apache Parquet, Apache ORC, or Delta Lake, is a cost-effective way to perform a change data capture (CDC) operation on semi-structured data stored in Amazon S3. An open source data lake format allows you to query data directly from S3 using standard SQL, without the need to move or copy data to another service. An open source data lake format also supports schema evolution, meaning it can handle changes in the data structure over time. An open source data lake format also supports upserts, meaning it can insert new data and update existing data in the same operation, using a merge command. This way, you can efficiently capture the changes from the data source and apply them to the S3 data lake, without duplicating or losing any data. The other options are not as cost-effective as using an open source data lake format, as they involve additional steps or costs. Option A requires you to create and maintain an AWS Lambda function, which can be complex and error-prone. AWS Lambda also has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the CDC operation. Option B and D require you to ingest the data into a relational database service, such as Amazon RDS or Amazon Aurora, which can be expensive and unnecessary for semi-structured data. AWS Database Migration Service (AWS DMS) can write the changed data to the data lake, but it alsocharges you for the data replication and transfer. Additionally, AWS DMS does not support JSON as a source data type, so you would need to convert the data to a supported format before using AWS DMS. References:
? What is a data lake?
? Choosing a data format for your data lake
? Using the MERGE INTO command in Delta Lake
? [AWS Lambda quotas]
? [AWS Database Migration Service quotas]

NEW QUESTION 8
A company created an extract, transform, and load (ETL) data pipeline in AWS Glue. A data engineer must crawl a table that is in Microsoft SQL Server. The data engineer needs to extract, transform, and load the output of the crawl to an Amazon S3 bucket. The data engineer also must orchestrate the data pipeline.
Which AWS service or feature will meet these requirements MOST cost-effectively?

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

Answer: B

Explanation:
AWS Glue workflows are a cost-effective way to orchestrate complex ETL jobs that involve multiple crawlers, jobs, and triggers. AWS Glue workflows allow you to visually monitor the progress and dependencies of your ETL tasks, and automatically handle errors and retries. AWS Glue workflows also integrate with other AWS services, such as Amazon S3, Amazon Redshift, and AWS Lambda, among others, enabling you to leverage these services for your data processing workflows. AWS Glue workflows are serverless, meaning you only pay for the resources you use, and you don't have to manage any infrastructure.
AWS Step Functions, AWS Glue Studio, and Amazon MWAA are also possible options for orchestrating ETL pipelines, but they have some drawbacks compared to AWS Glue workflows. AWS Step Functions is a serverless function orchestrator that can handle different types of data processing, such as real-time, batch, and stream processing. However, AWS Step Functions requires you to write code to define your state machines, which can be complex and error-prone. AWS Step Functions also charges you for every state transition, which can add up quickly for large-scale ETL pipelines.
AWS Glue Studio is a graphical interface that allows you to create and run AWS Glue ETL jobs without writing code. AWS Glue Studio simplifies the process of

building, debugging, and monitoring your ETL jobs, and provides a range of pre-built transformations and connectors. However, AWS Glue Studio does not support workflows, meaning you cannot orchestrate multiple ETL jobs or crawlers with dependencies and triggers. AWS Glue Studio also does not support streaming data sources or targets, which limits its use cases for real-time data processing.
Amazon MWAA is a fully managed service that makes it easy to run open-source versions of Apache Airflow on AWS and build workflows to run your ETL jobs and data pipelines. Amazon MWAA provides a familiar and flexible environment for data engineers who are familiar with Apache Airflow, and integrates with a range of AWS services such as Amazon EMR, AWS Glue, and AWS Step Functions. However, Amazon MWAA is not serverless, meaning you have to provision and pay for the resources you need, regardless of your usage. Amazon MWAA also requires you to write code to define your DAGs, which can be challenging and time-consuming for complex ETL pipelines. References:
? AWS Glue Workflows
? AWS Step Functions
? AWS Glue Studio
? Amazon MWAA
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 9
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 10
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 14
A data engineer must orchestrate a data pipeline that consists of one AWS Lambda function and one AWS Glue job. The solution must integrate with AWS services.
Which solution will meet these requirements with the LEAST management overhead?

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

Answer: A

Explanation:
AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows. You can use Step Functions to create state machines that define the sequence and logic of the tasks in your workflow. Step Functions supports various types of tasks, such as Lambda functions, AWS Glue jobs, Amazon EMR clusters, Amazon ECS tasks, etc. You can use Step Functions to monitor and troubleshoot your workflows, as well as to handle errors and retries.
Using an AWS Step Functions workflow that includes a state machine to run the Lambda function and then the AWS Glue job will meet the requirements with the least management overhead, as it leverages the serverless and managed capabilities of Step Functions. You do not need to write any code to orchestrate the tasks in your workflow, as you can use the Step Functions console or the AWS Serverless Application Model (AWS SAM) to define and deploy your state machine. You also do not need to provision or manage any servers or clusters, as Step Functions scales automatically based on the demand.
The other options are not as efficient as using an AWS Step Functions workflow. Using an Apache Airflow workflow that is deployed on an Amazon EC2 instance or on Amazon Elastic Kubernetes Service (Amazon EKS) will require more management overhead, as you will need to provision, configure, and maintain the EC2 instance or the EKS cluster, as well as the Airflow components. You will also need to write and maintain the Airflow DAGs to orchestrate the tasks in your workflow. Using an AWS Glue workflow to run the Lambda function and then the AWS Glue job will not work, as AWS Glue workflows only support AWS Glue jobs and crawlers as tasks, not Lambda functions. References:
? AWS Step Functions
? AWS Glue
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.3: AWS Step Functions

NEW QUESTION 16
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 17
During a security review, a company identified a vulnerability in an AWS Glue job. The company discovered that credentials to access an Amazon Redshift cluster were hard coded in the job script.
A data engineer must remediate the security vulnerability in the AWS Glue job. The solution must securely store the credentials.
Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

A. Store the credentials in the AWS Glue job parameters.
B. Store the credentials in a configuration file that is in an Amazon S3 bucket.
C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.
D. Store the credentials in AWS Secrets Manager.
E. Grant the AWS Glue job 1AM role access to the stored credentials.

Answer: DE

Explanation:
AWS Secrets Manager is a service that allows you to securely store and manage secrets, such as database credentials, API keys, passwords, etc. You can use Secrets Manager to encrypt, rotate, and audit your secrets, as well as to control access to them using fine-grained policies. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue 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).
Storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials will meet the requirements, as it will remediate the security vulnerability in the AWS Glue job and securely store the credentials. By using AWS Secrets Manager, you can avoid hard coding the credentials in the job script, which is a bad practice that exposes the credentials to unauthorized access or leakage. Instead, you can store the credentials as a secret in Secrets Manager and reference the secret name or ARN in the job script. You can also use Secrets Manager to encrypt thecredentials using AWS Key Management Service (AWS KMS), rotate the credentials automatically or on demand, and monitor the access to the credentials using AWS CloudTrail. By granting the AWS Glue job 1AM role access to the stored credentials, you can use the principle of least privilege to ensure that only the AWS Glue job can retrieve the credentials from Secrets Manager. You can also use resource-based or tag-based policies to further restrict the access to the credentials.
The other options are not as secure as storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials. Storing the credentials in the AWS Glue job parameters will not remediate the security vulnerability, as the job parameters are still visible in the AWS Glue console and API. Storing the credentials in a configuration file that is in an Amazon S3 bucket and accessing the credentials from the configuration file by using the AWS Glue job will not be as secure as using Secrets Manager, as the configuration file may not be encrypted or rotated, and the access to the file may not be audited or controlled. References:
? AWS Secrets Manager
? AWS Glue
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

NEW QUESTION 19

A data engineer must use AWS services to ingest a dataset into an Amazon S3 data lake. The data engineer profiles the dataset and discovers that the dataset contains personally identifiable information (PII). The data engineer must implement a solution to profile the dataset and obfuscate the PII.
Which solution will meet this requirement with the LEAST operational effort?

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

Answer: C

Explanation:
AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue Studio is a graphical interface that allows you to easily author, run, and monitor AWS Glue ETL jobs. AWS Glue Data Quality is a feature that enables you to validate, cleanse, and enrich your data using predefined or custom rules. AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows.
Using the Detect PII transform in AWS Glue Studio, you can automatically identify and label the PII in your dataset, such as names, addresses, phone numbers, email addresses, etc. You can then create a rule in AWS Glue Data Quality to obfuscate the PII, such as masking, hashing, or replacing the values with dummy data. You can also use other rules to validate and cleanse your data, such as checking for null values, duplicates, outliers, etc. You can then use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake. You can use AWS Glue DataBrew to visually explore and transform the data, AWS Glue crawlers to discover and catalog the data, and AWS Glue jobs to load the data into the S3 data lake.
This solution will meet the requirement with the least operational effort, as it leverages the serverless and managed capabilities of AWS Glue, AWS Glue Studio, AWS Glue Data Quality, and AWS Step Functions. You do not need to write any code to identify or obfuscate the PII, as you can use the built-in transforms and rules in AWS Glue Studio and AWS Glue Data Quality. You also do not need to provision or manage any servers or clusters, as AWS Glue and AWS Step Functions scale automatically based on the demand. The other options are not as efficient as using the Detect PII transform in AWS Glue Studio, creating a rule in AWS Glue Data Quality, and using an AWS Step Functions state machine. Using an Amazon Kinesis Data Firehose delivery stream to process the dataset, creating an AWS Lambda transform function to identify the PII, using an AWS SDK to obfuscate the PII, and setting the S3 data lake as the target for the delivery stream will require more operational effort, as you will need to write and maintain code to identifyand obfuscate the PII, as well as manage the Lambda function and its resources. Using the Detect PII transform in AWS Glue Studio to identify the PII, obfuscating the PII, and using an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake will not be as effective as creating a rule in AWS Glue Data Quality to obfuscate the PII, as you will need to manually obfuscate the PII after identifying it, which can be error-prone and time-consuming. Ingesting the dataset into Amazon DynamoDB, creating an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data, and using the same Lambda function to ingest the data into the S3 data lake will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. You will also incur additional costs and complexity by using DynamoDB as an intermediate data store, which may not be necessary for your use case. References:
? AWS Glue
? AWS Glue Studio
? AWS Glue Data Quality
? [AWS Step Functions]
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

NEW QUESTION 20
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 25

A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views.
Which solution will meet this requirement with the LEAST effort?

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

Answer: C

Explanation:
The query editor v2 in Amazon Redshift is a web-based tool that allows users to run SQL queries and scripts on Amazon Redshift clusters. The query editor v2 supports creating and managing materialized views, which are precomputed results of a query that can improve the performance of subsequent queries. The query editor v2 also supports scheduling queries to run at specified intervals, which can be used to refresh materialized views automatically. This solution requires the least effort, as it does not involve any additional services, coding, or configuration. The other solutions are more complex and require more operational overhead. Apache Airflow is an open-source platform for orchestrating workflows, which can be used to refresh materialized views, but it requires setting up and managing an Airflow environment, creating DAGs (directed acyclic graphs) to define the workflows, and integrating with Amazon Redshift. AWS Lambda is a serverless compute service that can run code in response to events, which can be used to refresh materialized views, but it requires creating and deploying Lambda functions, defining UDFs within Amazon Redshift, and triggering the functions using events or schedules. AWS Glue is a fully managed ETL service that can run jobs to transform and load data, which can be used to refresh materialized views, but it requires creating and configuring Glue jobs, defining Glue workflows to orchestrate the jobs, and scheduling the workflows using triggers. References:
? Query editor V2
? Working with materialized views
? Scheduling queries
? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 26
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 29
A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size. A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.
Which solution will meet this requirement with the LEAST operational effort?

A. Create and run an Apache Spark job in an AWS Glue noteboo
B. Configure the job to read the S3 file and calculate the number of distinct customers.
C. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 fil
D. Run SQL queries from Amazon Athena to calculate the number of distinct customers.
E. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.
F. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Answer: D

Explanation:
AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. You can use DataBrew to create recipes that define the steps to apply to your data, such as filtering, renaming, splitting, or aggregating columns. You can also use DataBrew to run jobs that execute the recipes on your data sources, such as Amazon S3, Amazon Redshift, or Amazon Aurora. DataBrew integrates with AWS Glue Data Catalog, which is a centralized metadata repository for your data assets1.
The solution that meets the requirement with the least operational effort is to use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers. This solution has the following advantages:
? It does not require you to write any code, as DataBrew provides a graphical user
interface that lets you explore, transform, and visualize your data. You can use DataBrewto concatenate the columns that contain customer first names and last

names, and then use the COUNT_DISTINCT aggregate function to count the number of unique values in the resulting column2.
? It does not require you to provision, manage, or scale any servers, clusters, or notebooks, as DataBrew is a fully managed service that handles all the infrastructure for you. DataBrew can automatically scale up or down the compute resources based on the size and complexity of your data and recipes1.
? It does not require you to create or update any AWS Glue Data Catalog entries, as
DataBrew can automatically create and register the data sources and targets in the Data Catalog. DataBrew can also use the existing Data Catalog entries to access the data in S3 or other sources3.
Option A is incorrect because it suggests creating and running an Apache Spark job in an AWS Glue notebook. This solution has the following disadvantages:
? It requires you to write code, as AWS Glue notebooks are interactive development environments that allow you to write, test, and debug Apache Spark code using Python or Scala. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.
? It requires you to provision and manage a development endpoint, which is a serverless Apache Spark environment that you can connect to your notebook. You need to specify the type and number of workers for your development endpoint, and monitor its status and metrics.
? It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.
Option B is incorrect because it suggests creating an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file, and running SQL queries from Amazon Athena to calculate the number of distinct customers. This solution has the following disadvantages:
? It requires you to create and run a crawler, which is a program that connects to your data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the Data Catalog. You need to specify the data store, the IAM role, the schedule, and the output database for your crawler.
? It requires you to write SQL queries, as Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. You need to use Athena to concatenate the columns that contain customer first names and last names, and then use the COUNT(DISTINCT) aggregate function to count the number of unique values in the resulting column.
Option C is incorrect because it suggests creating and running an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers. This solution has the following disadvantages:
? It requires you to write code, as Amazon EMR Serverless is a service that allows you to run Apache Spark jobs on AWS without provisioning or managing any infrastructure. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.
? It requires you to create and manage an Amazon EMR Serverless cluster, which is a fully managed and scalable Spark environment that runs on AWS Fargate. You need to specify the cluster name, the IAM role, the VPC, and the subnet for your cluster, and monitor its status and metrics.
? It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.
References:
? 1: AWS Glue DataBrew - Features
? 2: Working with recipes - AWS Glue DataBrew
? 3: Working with data sources and data targets - AWS Glue DataBrew
? [4]: AWS Glue notebooks - AWS Glue
? [5]: Development endpoints - AWS Glue
? [6]: Populating the AWS Glue Data Catalog - AWS Glue
? [7]: Crawlers - AWS Glue
? [8]: Amazon Athena - Features
? [9]: Amazon EMR Serverless - Features
? [10]: Creating an Amazon EMR Serverless cluster - Amazon EMR
? [11]: Using the AWS Glue Data Catalog with Amazon EMR Serverless - Amazon EMR

NEW QUESTION 30
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 34
A company uses an on-premises Microsoft SQL Server database to store financial transaction data. The company migrates the transaction data from the on-premises database to AWS at the end of each month. The company has noticed that the cost to migrate data from the on-premises database to an Amazon RDS for SQL Server database has increased recently.
The company requires a cost-effective solution to migrate the data to AWS. The solution must cause minimal downtown for the applications that access the database.
Which AWS service should the company use to meet these requirements?

A. AWS Lambda
B. AWS Database Migration Service (AWS DMS)
C. AWS Direct Connect
D. AWS DataSync

Answer: B

Explanation:
AWS Database Migration Service (AWS DMS) is a cloud service that makes it possible to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores to AWS quickly, securely, and with minimal downtime and zero data loss1. AWS DMS supports migration between 20-plus database and analytics engines, such as Microsoft SQL Server to Amazon RDS for SQL Server2. AWS DMS takes overmany of the difficult or tedious tasks involved in a migration project, such as capacity analysis, hardware and software procurement, installation and administration, testing and debugging, and ongoing replication and monitoring1. AWS DMS is a cost-effective solution, as you only pay for the compute resources and additional log storage used during the migration process2. AWS DMS is the best solution for the company to migrate the financial transaction data from the on-premises Microsoft SQL Server database to AWS, as it meets the requirements of minimal downtime, zero data loss, and low cost.
Option A is not the best solution, as AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, but it does not provide any built-in features for database migration. You would have to write your own code to extract, transform, and load the data from the source to the target, which would increase the operational overhead and complexity.
Option C is not the best solution, as AWS Direct Connect is a service that establishes a dedicated network connection from your premises to AWS, but it does not provide any built-in features for database migration. You would still need to use another service or tool to perform the actual data transfer, which would increase the cost and complexity.
Option D is not the best solution, as AWS DataSync is a service that makes it easy to transfer data between on-premises storage systems and AWS storage services, such as Amazon S3, Amazon EFS, and Amazon FSx for Windows File Server, but it does not support Amazon RDS for SQL Server as a target. You would have to use another service or tool to migrate the data from Amazon S3 to Amazon RDS for SQL Server, which would increase the latency and complexity.
References:
? Database Migration - AWS Database Migration Service - AWS
? What is AWS Database Migration Service?
? AWS Database Migration Service Documentation
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 36
A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform.
The company wants to minimize the effort and time required to incorporate third-party datasets.
Which solution will meet these requirements with the LEAST operational overhead?

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

Answer: A

Explanation:
AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed12.
The other options are not optimal for the following reasons:
? B. Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and loading (ETL) jobs, which can add operational overhead3.
? C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, suchas clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data .
? D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR). This option is also not feasible, as Amazon ECR is a fully managed container registry service that stores, manages, and deploys container images, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams does not support accessing and integrating data from Amazon ECR, which is meant for storing and managing container images, not data .
References:
? 1: AWS Data Exchange User Guide
? 2: AWS Data Exchange FAQs
? 3: AWS Glue Developer Guide
? : AWS CodeCommit User Guide
? : Amazon Kinesis Data Streams Developer Guide
? : Amazon Elastic Container Registry User Guide
? : Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source

NEW QUESTION 41
A data engineering team is using an Amazon Redshift data warehouse for operational reporting. The team wants to prevent performance issues that might result from long- running queries. A data engineer must choose a system table in Amazon Redshift to record anomalies when a query optimizer identifies conditions that might indicate performance issues.
Which table views should the data engineer use to meet this requirement?

A. STL USAGE CONTROL
B. STL ALERT EVENT LOG
C. STL QUERY METRICS
D. STL PLAN INFO

Answer: B

Explanation:
The STL ALERT EVENT LOG table view records anomalies when the query optimizer identifies conditions that might indicate performance issues. These conditions include skewed data distribution, missing statistics, nested loop joins, and broadcasted data. The STL ALERT EVENT LOG table view can help the data engineer to identify and troubleshoot the root causes of performance issues and optimize the query execution plan. The other table views are not relevant for this requirement. STL USAGE CONTROL records the usage limits and quotas for Amazon Redshift resources. STL QUERY METRICS records the execution time and resource consumption of queries. STL PLAN INFO records the query execution plan and the steps involved in each query. References:
? STL ALERT EVENT LOG
? System Tables and Views
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 44
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 45
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 49
A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during non-business hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to comply with company policies.
A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.

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

A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day
B. Use the query result reuse feature of Amazon Athena for the SQL queries.
C. Add an Amazon ElastiCache cluster between the Bl application and Athena.
D. Change the format of the files that are in the dataset to Apache Parquet.

Answer: B

Explanation:
The best solution to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs is to use the query result reuse feature of AmazonAthena for the SQL queries. This feature allows you to run the same query multiple times without incurring additional charges, as long as the underlying data has not changed and the query results are still in the query result location in Amazon S31. This feature is useful for scenarios where you have a petabyte-scale dataset that is updated infrequently, such as once a day, and you have a BI application that runs the same queries repeatedly, such as every hour. By using the query result reuse feature, you can reduce the amount of data scanned by your queries and save on the cost of running Athena. You can enable or disable this feature at the workgroup level or at the individual query level1.
Option A is not the best solution, as configuring an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon S3 Lifecycle policies are rules that you can define to automatically transition objects between different storage classes based on specified criteria, such as the age of the object2. S3 Glacier Deep Archive is the lowest-cost storage class in Amazon S3, designed
for long-term data archiving that is accessed once or twice in a year3. While moving data to S3 Glacier Deep Archive can reduce the storage cost, it would also increase the retrieval cost and latency, as it takes up to 12 hours to restore the data from S3 Glacier Deep Archive3. Moreover, Athena does not support querying data that is in S3 Glacier or S3 Glacier Deep Archive storage classes4. Therefore, using this option would not meet the requirements of running on-demand SQL queries on the dataset.
Option C is not the best solution, as adding an Amazon ElastiCache cluster between the BI application and Athena would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon ElastiCache is a service that offers fully managed in-memory data stores, such as Redis and Memcached, that can improve the performance and scalability of web applications by caching frequently accessed data. While using ElastiCache can reduce the latency and load on the BI application, it would not reduce the amount of data scanned by Athena, which is the main factor that determines the cost of running Athena. Moreover, using ElastiCache would introduce additional infrastructure costs and operational overhead, as you would have to provision, manage, and scale the ElastiCache cluster, and integrate it with the BI application and Athena. Option D is not the best solution, as changing the format of the files that are in the dataset to Apache Parquet would not cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs, but rather increase the complexity. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However,changing the format of the files that are in the dataset to Apache Parquet would require additional processing and transformation steps, such as using AWS Glue or Amazon EMR to convert the files from their original format to Parquet, and storing the converted files in a separate location in Amazon S3. This would increase the complexity and the operational overhead of the data pipeline, and also incur additional costs for using AWS Glue or Amazon EMR. References:
? Query result reuse
? Amazon S3 Lifecycle
? S3 Glacier Deep Archive
? Storage classes supported by Athena
? [What is Amazon ElastiCache?]
? [Amazon Athena pricing]
? [Columnar Storage Formats]
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 52
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 55
A company is developing an application that runs on Amazon EC2 instances. Currently, the data that the application generates is temporary. However, the company needs to persist the data, even if the EC2 instances are terminated.
A data engineer must launch new EC2 instances from an Amazon Machine Image (AMI) and configure the instances to preserve the data.
Which solution will meet this requirement?

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

Answer: C

Explanation:
Amazon EC2 instances can use two types of storage volumes: instance store volumes and Amazon EBS volumes. Instance store volumes are ephemeral, meaning they are only attached to the instance for the duration of its life cycle. If the instance is stopped, terminated, or fails, the data on the instance store volume is lost. Amazon EBS volumes are persistent, meaning they can be detached from the instance and attached to another instance, and the data on the volume is preserved. To meet the requirement of persisting the data even if the EC2 instances are terminated, the data engineer must use Amazon EBS volumes to store the application data. The solution is to launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume, which is the default option for most AMIs. Then, the data engineer must attach an Amazon EBS volume to each instance and configure the application to write the data to the EBS volume. This way, the data will be saved on the EBS volume and can be accessed by another instance if needed. The data engineer can apply the default settings to the EC2 instances, as there is no need to modify the instance type, security group, or IAM role for this solution. The other options are either not feasible or not optimal. Launching new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data (option A) or by using an AMI that is backed by a root Amazon EBS volume that contains the application data (option B) would not work, as the data on the AMI would be outdated and overwritten by the new instances. Attaching an additional EC2 instance store volume to contain the application data (option D)would not work, as the data on the instance store volume would be lost if the instance is terminated. References:
? Amazon EC2 Instance Store
? Amazon EBS Volumes
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.1: Amazon EC2

NEW QUESTION 58
A company is migrating a legacy application to an Amazon S3 based data lake. A data engineer reviewed data that is associated with the legacy application. The data engineer found that the legacy data contained some duplicate information.
The data engineer must identify and remove duplicate information from the legacy application data.
Which solution will meet these requirements with the LEAST operational overhead?

A. Write a custom extract, transform, and load (ETL) job in Pytho
B. Use the DataFramedrop duplicatesf) function by importingthe Pandas library to perform data deduplication.
C. Write an AWS Glue extract, transform, and load (ETL) jo
D. Usethe FindMatches machine learning(ML) transform to transform the data to perform data deduplication.
E. Write a custom extract, transform, and load (ETL) job in Pytho
F. Import the Python dedupe librar
G. Use the dedupe library to perform data deduplication.
H. Write an AWS Glue extract, transform, and load (ETL) jo
I. Import the Python dedupe librar
J. Use the dedupe library to perform data deduplication.

Answer: B

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle data deduplication with minimal operational overhead. AWS Glue provides a built-in ML transform called FindMatches, which can automatically identify and group similar records in a dataset. FindMatches can also generate a primary key for each group of records and remove duplicates. FindMatches does not require any coding or prior ML experience, as it can learn from a sample of labeled data provided by the user. FindMatches can also scale to handle large datasets and optimize the cost and performance of the ETL job. References:
? AWS Glue
? FindMatches ML Transform
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 62
A healthcare company uses Amazon Kinesis Data Streams to stream real-time health data from wearable devices, hospital equipment, and patient records.
A data engineer needs to find a solution to process the streaming data. The data engineer needs to store the data in an Amazon Redshift Serverless warehouse. The solution must support near real-time analytics of the streaming data and the previous day's data. Which solution will meet these requirements with the LEAST operational overhead?

A. Load data into Amazon Kinesis Data Firehos
B. Load the data into Amazon Redshift.

C. Use the streaming ingestion feature of Amazon Redshift.
D. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.
E. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Answer: B

Explanation:
The streaming ingestion feature of Amazon Redshift enables you to ingest data from streaming sources, such as Amazon Kinesis Data Streams, into Amazon Redshift tables in near real-time. You can use the streaming ingestion feature to process the streaming data from the wearable devices, hospital equipment, and patient records. The streaming ingestion feature also supports incremental updates, which means you can append new data or update existing data in the Amazon Redshift tables. This way, you can store the data in an Amazon Redshift Serverless warehouse and support near real-time analytics of the streaming data and the previous day's data. This solution meets the requirements with the least operational overhead, as it does not require any additional services or components to ingest and process the streaming data. The other options are either not feasible or not optimal. Loading data into Amazon Kinesis Data Firehose and then into Amazon Redshift (option A) would introduce additional latency and cost, as well as require additional configuration and management. Loading data into Amazon S3 and then using the COPY command to load the data into Amazon Redshift (option C) would also introduce additional latency and cost, as well as require additional storage space and ETL logic. Using the Amazon Aurora zero-ETL integration with Amazon Redshift (option D) would not work, as it requires the data to be stored in Amazon Aurora first, which is not the case for the streaming data from the healthcare company. References:
? Using streaming ingestion with Amazon Redshift
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.5: Amazon Redshift Streaming Ingestion

NEW QUESTION 67
A company uses Amazon RDS to store transactional data. The company runs an RDS DB instance in a private subnet. A developer wrote an AWS Lambda function with default settings to insert, update, or delete data in the DB instance.
The developer needs to give the Lambda function the ability to connect to the DB instance privately without using the public internet.
Which combination of steps will meet this requirement with the LEAST operational overhead? (Choose two.)

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

Answer: CD

Explanation:
To enable the Lambda function to connect to the RDS DB instance privately without using the public internet, the best combination of steps is to configure the Lambda function to run in the same subnet that the DB instance uses, and attach the same security group to the Lambda function and the DB instance. This way, the Lambda function and the DB instance can communicate within the same private network, and the security group can allow traffic between them on the database port. This solution has the least operational overhead, as it does not require any changes to the public access setting, the network ACL, or the security group of the DB instance.
The other options are not optimal for the following reasons:
? A. Turn on the public access setting for the DB instance. This option is not recommended, as it would expose the DB instance to the public internet, which can compromise the security and privacy of the data. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.
? B. Update the security group of the DB instance to allow only Lambda function invocations on the database port. This option is not sufficient, as it would only modify the inbound rules of the security group of the DB instance, but not the outbound rules of the security group of the Lambda function. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.
? E. Update the network ACL of the private subnet to include a self-referencing rule
that allows access through the database port. This option is not necessary, as the network ACL of the private subnet already allows all traffic within the subnet by default. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.
References:
? 1: Connecting to an Amazon RDS DB instance
? 2: Configuring a Lambda function to access resources in a VPC
? 3: Working with security groups
? : Network ACLs

NEW QUESTION 70
A company currently stores all of its data in Amazon S3 by using the S3 Standard storage class.
A data engineer examined data access patterns to identify trends. During the first 6 months, most data files are accessed several times each day. Between 6 months and 2 years, most data files are accessed once or twice each month. After 2 years, data files are accessed only once or twice each year.
The data engineer needs to use an S3 Lifecycle policy to develop new data storage rules. The new storage solution must continue to provide high availability.
Which solution will meet these requirements in the MOST cost-effective way?

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

Answer: C

Explanation:
To achieve the most cost-effective storage solution, the data engineer needs to use an S3 Lifecycle policy that transitions objects to lower-cost storage classes based on their access patterns, and deletes them when they are no longer needed. The storage classes should also provide high availability, which means they

should be resilient to the loss of data in a single Availability Zone1. Therefore, the solution must include the following steps:
? Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. S3 Standard-IA is designed for data that is accessed less frequently, but requires rapid access when needed. It offers the same high durability, throughput, and low latency as S3 Standard, but with a lower storage cost and a retrieval fee2.
Therefore, it is suitablefor data files that are accessed once or twice each month. S3 Standard-IA also provides high availability, as it stores data redundantly across multiple Availability Zones1.
? Transfer objects to S3 Glacier Deep Archive after 2 years. S3 Glacier Deep Archive is the lowest-cost storage class that offers secure and durable storage for data that is rarely accessed and can tolerate a 12-hour retrieval time. It is ideal for long-term archiving and digital preservation3. Therefore, it is suitable for data files that are accessed only once or twice each year. S3 Glacier Deep Archive also provides high availability, as it stores data across at least three geographically dispersed Availability Zones1.
? Delete objects when they are no longer needed. The data engineer can specify an expiration action in the S3 Lifecycle policy to delete objects after a certain period of time. This will reduce the storage cost and comply with any data retention policies.
Option C is the only solution that includes all these steps. Therefore, option C is the correct answer.
Option A is incorrect because it transitions objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. S3 One Zone-IA is similar to S3 Standard-IA, but it stores data in a single Availability Zone. This means it has a lower availability and durability than S3 Standard-IA, and it is not resilient to the loss of data in a single Availability Zone1. Therefore, it does not provide high availability as required.
Option B is incorrect because it transfers objects to S3 Glacier Flexible Retrieval after 2 years. S3 Glacier Flexible Retrieval is a storage class that offers secure and durable storage for data that is accessed infrequently and can tolerate a retrieval time of minutes to hours. It is more expensive than S3 Glacier Deep Archive, and it is not suitable for data that is accessed only once or twice each year3. Therefore, it is not the most cost-effective option.
Option D is incorrect because it combines the errors of option A and B. It transitions objects to S3 One Zone-IA after 6 months, which does not provide high availability, and it transfers objects to S3 Glacier Flexible Retrieval after 2 years, which is not the most cost-effective option.
References:
? 1: Amazon S3 storage classes - Amazon Simple Storage Service
? 2: Amazon S3 Standard-Infrequent Access (S3 Standard-IA) - Amazon Simple Storage Service
? 3: Amazon S3 Glacier and S3 Glacier Deep Archive - Amazon Simple Storage Service
? [4]: Expiring objects - Amazon Simple Storage Service
? [5]: Managing your storage lifecycle - Amazon Simple Storage Service
? [6]: Examples of S3 Lifecycle configuration - Amazon Simple Storage Service
? [7]: Amazon S3 Lifecycle further optimizes storage cost savings with new features
- What's New with AWS

NEW QUESTION 71
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 76
A data engineer runs Amazon Athena queries on data that is in an Amazon S3 bucket. The Athena queries use AWS Glue Data Catalog as a metadata table. The data engineer notices that the Athena query plans are experiencing a performance bottleneck. The data engineer determines that the cause of the performance bottleneck is the large number of partitions that are in the S3 bucket. The data engineer must resolve the performance bottleneck and reduce Athena query planning time.
Which solutions will meet these requirements? (Choose two.)

A. Create an AWS Glue partition inde
B. Enable partition filtering.
C. Bucketthe data based on a column thatthe data have in common in a WHERE clause of the user query
D. Use Athena partition projection based on the S3 bucket prefix.

E. Transform the data that is in the S3 bucket to Apache Parquet format.
F. Use the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects.

Answer: AC

Explanation:
The best solutions to resolve the performance bottleneck and reduce Athena query planning time are to create an AWS Glue partition index and enable partition filtering, and to use Athena partition projection based on the S3 bucket prefix.
AWS Glue partition indexes are a feature that allows you to speed up query processing of highly partitioned tables cataloged in AWS Glue Data Catalog. Partition indexes are available for queries in Amazon EMR, Amazon Redshift Spectrum, and AWS Glue ETL jobs. Partition indexes are sublists of partition keys defined in the table. When you create a partition index, you specify a list of partition keys that already exist on a given table. AWS Glue then creates an index for the specified keys and stores it in the Data Catalog. When you run a query that filters on the partition keys, AWS Glue uses the partition index to quickly identify the relevant partitions without scanning the entiretable metadata. This reduces the query planning time and improves the query performance1.
Athena partition projection is a feature that allows you to speed up query processing of highly partitioned tables and automate partition management. In partition projection, Athena calculates partition values and locations using the table properties that you configure directly on your table in AWS Glue. The table properties allow Athena to 'project', or determine, the necessary partition information instead of having to do a more time- consuming metadata lookup in the AWS Glue Data Catalog. Because in-memory operations are often faster than remote operations, partition projection can reduce the runtime of queries against highly partitioned tables. Partition projection also automates partition management because it removes the need to manually create partitions in Athena, AWS Glue, or your external Hive metastore2.
Option B is not the best solution, as bucketing the data based on a column that the data have in common in a WHERE clause of the user query would not reduce the query planning time. Bucketing is a technique that divides data into buckets based on a hash function applied to a column. Bucketing can improve the performance of join queries by
reducing the amount of data that needs to be shuffled between nodes. However, bucketing does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario3.
Option D is not the best solution, as transforming the data that is in the S3 bucket to Apache Parquet format would not reduce the query planning time. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However, Parquet does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario4.
Option E is not the best solution, as using the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects would not reduce the query planning time. S3DistCP is a tool that can copy large amounts of data between Amazon S3 buckets or from HDFS to Amazon S3. S3DistCP can also aggregate smaller files into larger files to improve the performance of sequential access. However, S3DistCP does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario5. References:
? Improve query performance using AWS Glue partition indexes
? Partition projection with Amazon Athena
? Bucketing vs Partitioning
? Columnar Storage Formats
? S3DistCp
? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 81
......

Relate Links

100% Pass Your AWS-Certified-Data-Engineer-Associate Exam with Exambible Prep Materials

https://www.exambible.com/AWS-Certified-Data-Engineer-Associate-exam/

Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - https://www.exambible.com/