

AWS-Certified-Data-Engineer-Associate Dumps

AWS Certified Data Engineer - Associate (DEA-C01)

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



NEW QUESTION 1

A data engineer 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 E also does not support querying 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 company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account.

Which solution will meet these requirements?

- A. Create an S3 bucket for each use cas
- B. Create an S3 bucket policy that grants permissions to appropriate individual IAM user
- C. Apply the S3 bucket policy to the S3 bucket.
- D. Create an Athena workgroup for each use cas
- E. Apply tags to the workgrou
- F. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.
- G. Create an IAM role for each use cas
- H. Assign appropriate permissions to the role for each use cas
- I. Associate the role with Athena.
- J. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use cas
- K. Apply the resource policy to the specific tables that Athena uses.

Answer: B

Explanation:

Athena workgroups are a way to isolate query execution and query history among users, teams, and applications that share the same AWS account. By creating a workgroup for each use case, the company can control the access and actions on the workgroup resource using resource-level IAM permissions or identity-based IAM policies. The company can also use tags to organize and identify the workgroups, and use them as conditions in the IAM policies to grant or deny permissions to the workgroup. This solution meets the requirements of separating query processes and access to query history among users, teams, and applications that are in the same AWS account. References:

? Athena Workgroups

? IAM policies for accessing workgroups

? Workgroup example policies

NEW QUESTION 3

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 Amazon Redshift. 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 4

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

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

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

Answer: A

Explanation:

The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). References:

? Introducing AWS Glue Flex jobs: Cost savings on ETL workloads

? Serverless Data Integration – AWS Glue Pricing

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

NEW QUESTION 5

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

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

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

Answer: A

Explanation:

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

? Amazon Managed Service for Apache Flink

? Apache Flink

? Window Aggregations in Flink

NEW QUESTION 6

A company receives .csv files that contain physical address data. The data is in columns that have the following names: Door_No, Street_Name, City, and Zip_Code. The company wants to create a single column to store these values in the following format:

```
{
  "Door_No": "24",
  "Street_Name": "AAA street",
  "City": "BBB",
  "Zip_Code": "111111"
}
```

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

- A. Use AWS Glue DataBrew to read the file
- B. Use the NEST TO ARRAY transformation to create the new column.
- C. Use AWS Glue DataBrew to read the file
- D. Use the NEST TO MAP transformation to create the new column.
- E. Use AWS Glue DataBrew to read the file
- F. Use the PIVOT transformation to create the new column.
- G. Write a Lambda function in Python to read the file
- H. Use the Python data dictionary type to create the new column.

Answer: B

Explanation:

The NEST TO MAP transformation allows you to combine multiple columns into a single column that contains a JSON object with key-value pairs. This is the easiest way to achieve the desired format for the physical address data, as you can simply select the columns to nest and specify the keys for each column. The NEST TO ARRAY transformation creates a single column that contains an array of values, which is not the same as the JSON object format. The PIVOT

transformation reshapes the data by creating new columns from unique values in a selected column, which is not applicable for this use case. Writing a Lambda function in Python requires more coding effort than using AWS Glue DataBrew, which provides a visual and interactive interface for data transformations.

References:

? 7 most common data preparation transformations in AWS Glue DataBrew (Section: Nesting and unnesting columns)

? NEST TO MAP - AWS Glue DataBrew (Section: Syntax)

NEW QUESTION 7

A data engineer is building a data pipeline on AWS by using AWS Glue extract, transform, and load (ETL) jobs. The data engineer needs to process data from Amazon RDS and MongoDB, perform transformations, and load the transformed data into Amazon Redshift for analytics. The data updates must occur every hour. Which combination of tasks will meet these requirements with the LEAST operational overhead? (Choose two.)

- A. Configure AWS Glue triggers to run the ETL jobs even/ hour.
- B. Use AWS Glue DataBrew to clean and prepare the data for analytics.
- C. Use AWS Lambda functions to schedule and run the ETL jobs even/ hour.
- D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.
- E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Answer: AD

Explanation:

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

? AWS Glue triggers

? AWS Glue connections

? AWS Glue DataBrew

? [AWS Lambda functions]

? [Redshift Data API]

NEW QUESTION 8

A data engineer needs to use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket. When the data engineer connects to the QuickSight dashboard, the data engineer receives an error message that indicates insufficient permissions. Which factors could cause the permissions-related errors? (Choose two.)

- A. There is no connection between QuickSight and Athena.
- B. The Athena tables are not cataloged.
- C. QuickSight does not have access to the S3 bucket.
- D. QuickSight does not have access to decrypt S3 data.
- E. There is no IAM role assigned to QuickSight.

Answer: CD

Explanation:

QuickSight does not have access to the S3 bucket and QuickSight does not have access to decrypt S3 data are two possible factors that could cause the permissions-related errors. Amazon QuickSight is a business intelligence service that allows you to create and share interactive dashboards based on various data sources, including Amazon Athena. Amazon Athena is a serverless query service that allows you to analyze data stored in Amazon S3 using standard SQL. To use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket, you need to grant QuickSight access to both Athena and S3, as well as any encryption keys that are used to encrypt the S3 data. If QuickSight does not have access to the S3 bucket or the encryption keys, it will not be able to read the data from Athena and display it on the dashboard, resulting in an error message that indicates insufficient permissions.

The other options are not factors that could cause the permissions-related errors. Option A, there is no connection between QuickSight and Athena, is not a factor, as QuickSight supports Athena as a native data source, and you can easily create a connection between them using the QuickSight console or the API. Option B, the Athena tables are not cataloged, is not a factor, as QuickSight can automatically discover the Athena tables that are cataloged in the AWS Glue Data Catalog, and you can also manually specify the Athena tables that are not cataloged. Option E, there is no IAM role assigned to QuickSight, is not a factor, as QuickSight requires an IAM role to access any AWS data sources, including Athena and S3, and you can create and assign an IAM role to QuickSight using the QuickSight console or the API. References:

? Using Amazon Athena as a Data Source

? Granting Amazon QuickSight Access to AWS Resources

? Encrypting Data at Rest in Amazon S3

NEW QUESTION 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 data
- B. Store the data in Amazon S3 for querying.
- C. Use AWS Lambda to process the sensor data
- D. Store the data in Amazon S3 for querying.
- E. Use Amazon Kinesis Data Streams to capture the sensor data
- F. Store the data in Amazon DynamoDB for querying.
- G. Use Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data
- H. Use AWS Glue to store the data 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 capture changes 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 needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3.

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

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

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

Answer: B

Explanation:

This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various

AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions¹².

The other options are not optimal for the following reasons:

? A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically. This option is also not recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog.

References:

? 1: AWS Glue Data Catalog

? 2: AWS Glue Crawlers

? : Amazon Aurora

? : AWS Lambda

? : Amazon DynamoDB

NEW QUESTION 10

A 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 11

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

Which solution will meet these requirements MOST cost-effectively?

- A. Create an AWS Glue Data Catalog
- B. Configure an AWS Glue Schema Registry
- 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 cluster
- 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 workgroup
- G. Explore the data that is in Amazon S3 by using Apache Spark through Athena
- H. Provide the Athena workgroup schema and tables to the analytics department.
- I. Create an AWS Glue Data Catalog
- J. Configure an AWS Glue Schema Registry
- K. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API
- 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 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 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 made^{6,7}. 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 and manage 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 made^{6,7}. 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 16

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

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

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

- A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization
- B. Optimize the problematic queries.
- C. Modify the database schema to include additional tables and indexes.
- D. Reboot the RDS DB instance once each week.
- E. Upgrade to a larger instance size.
- F. Implement caching to reduce the database query load.

Answer: AE

Explanation:

Amazon RDS is a fully managed service that provides relational databases in the cloud. Amazon RDS for MySQL is one of the supported database engines that you can use to run your applications. Amazon RDS provides various features and tools to monitor and optimize the performance of your DB instances, such as Performance Insights, Enhanced Monitoring, CloudWatch metrics and alarms, etc.

Using the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization and optimizing the problematic queries will help reduce the CPU utilization of the DB instance. Performance Insights is a feature that allows you to analyze the load on your DB instance and determine what is causing performance issues. Performance Insights collects, analyzes, and displays database performance data using an interactive dashboard. You can use Performance Insights to identify the top SQL statements, hosts, users, or processes that are consuming the most CPU resources. You can also drill down into the details of each query and see the execution plan, wait events, locks, etc. By using Performance Insights, you can pinpoint the root cause of the high CPU utilization and optimize the queries accordingly. For example, you can rewrite the queries to make them more efficient, add or remove indexes, use prepared statements, etc. Implementing caching to reduce the database query load will also help reduce the CPU utilization of the DB instance. Caching is a technique that allows you to store frequently accessed data in a fast and scalable storage layer, such as Amazon ElastiCache. By using caching, you can reduce the number of requests that hit your database, which in turn reduces the CPU load on your DB instance. Caching also improves the performance and availability of your application, as it reduces the latency and increases the throughput of your data access. You can use caching for various scenarios, such as storing session data, user preferences, application configuration, etc. You can also use caching for read-heavy workloads, such as displaying product details, recommendations, reviews, etc.

The other options are not as effective as using Performance Insights and caching. Modifying the database schema to include additional tables and indexes may or may not improve the CPU utilization, depending on the nature of the workload and the queries. Adding more tables and indexes may increase the complexity and overhead of the database, which may negatively affect the performance. Rebooting the RDS DB instance once each week will not reduce the CPU utilization, as it will not address the underlying cause of the high CPU load. Rebooting may also cause downtime and disruption to your application. Upgrading to a larger instance size may reduce the CPU utilization, but it will also increase the cost and complexity of your solution. Upgrading may also not be necessary if you can optimize the queries and reduce the database load by using caching. References:

? Amazon RDS

? Performance Insights

? Amazon ElastiCache

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 3: Data Storage and Management, Section 3.1: Amazon RDS

NEW QUESTION 17

A data engineer must orchestrate a series of Amazon Athena queries that will run every day. Each query can run for more than 15 minutes.

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

- A. Use an AWS Lambda function and the Athena Boto3 client `start_query_execution` API call to invoke the Athena queries programmatically.
- B. Create an AWS Step Functions workflow and add two state
- C. Add the first state before the Lambda function
- D. Configure the second state as a Wait state to periodically check whether the Athena query has finished using the Athena Boto3 `get_query_execution` API call
- E. Configure the workflow to invoke the next query when the current query has finished running.
- F. Use an AWS Glue Python shell job and the Athena Boto3 client `start_query_execution` API call to invoke the Athena queries programmatically.
- G. Use an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully
- H. Configure the Python shell script to invoke the next query when the current query has finished running.
- I. Use Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch.

Answer: AB

Explanation:

Option A and B are the correct answers because they meet the requirements most cost-effectively. Using an AWS Lambda function and the Athena Boto3 client `start_query_execution` API call to invoke the Athena queries programmatically is a simple and scalable way to orchestrate the queries. Creating an AWS Step Functions workflow and adding two states to check the query status and invoke the next query is a reliable and efficient way to handle the long-running queries.

Option C is incorrect because using an AWS Glue Python shell job to invoke the Athena queries programmatically is more expensive than using a Lambda function, as it requires provisioning and running a Glue job for each query.

Option D is incorrect because using an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully is not a cost-effective or reliable way to orchestrate the queries, as it wastes resources and time.

Option E is incorrect because using Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch is an overkill solution that introduces unnecessary complexity and cost, as it requires setting up and managing an Airflow environment and an AWS Batch compute environment.

References:

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.2: AWS Lambda, Section 5.3: AWS Step Functions, Pages 125-135

? Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.1: AWS Lambda, Lesson 5.2: AWS Step Functions, Pages 1-15

? AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon Athena, Pages 1-4

? AWS Documentation Overview, AWS Step Functions Developer Guide, Getting Started, Tutorial: Create a Hello World Workflow, Pages 1-8

NEW QUESTION 18

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 21

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

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

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

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

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

D. Create an AWS Lambda function

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

Answer: B

Explanation:

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. You can use Lambda to create functions that perform custom logic and integrate with other AWS services, such as API Gateway. Lambda automatically scales your application by running code in response to each trigger. You pay only for the compute time you consume¹.

Amazon ECS is a fully managed container orchestration service that allows you to run and scale containerized applications on AWS. You can use ECS to deploy, manage, and scale Docker containers using either Amazon EC2 instances or AWS Fargate, a serverless compute engine for containers².

Amazon EKS is a fully managed Kubernetes service that allows you to run Kubernetes clusters on AWS without needing to install, operate, or maintain your own Kubernetes control plane. You can use EKS to deploy, manage, and scale containerized applications using Kubernetes on AWS³.

The solution that meets the requirements with the least operational overhead is to create an AWS Lambda Python function with provisioned concurrency. This solution has the following advantages:

? It does not require you to provision, manage, or scale any servers or clusters, as Lambda handles all the infrastructure for you. This reduces the operational complexity and cost of running your code.

? It allows you to write your Python script as a Lambda function and integrate it with API Gateway using a simple configuration. API Gateway can invoke your Lambda function synchronously or asynchronously, and return the results to the frontend website.

? It ensures that your Lambda function is ready to respond to API requests without any cold start delays, by using provisioned concurrency. Provisioned concurrency is a feature that keeps your function initialized and hyper-ready to respond in double-digit milliseconds. You can specify the number of concurrent executions that you want to provision for your function.

Option A is incorrect because it requires you to deploy a custom Python script on an Amazon ECS cluster. This solution has the following disadvantages:

? It requires you to provision, manage, and scale your own ECS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

? It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process.

? It requires you to configure your ECS cluster to integrate with API Gateway, either using an Application Load Balancer or a Network Load Balancer. This adds another layer of complexity to your architecture.

Option C is incorrect because it requires you to deploy a custom Python script that can integrate with API Gateway on Amazon EKS. This solution has the following disadvantages:

? It requires you to provision, manage, and scale your own EKS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

? It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process.

? It requires you to configure your EKS cluster to integrate with API Gateway, either using an Application Load Balancer, a Network Load Balancer, or a service of type LoadBalancer. This adds another layer of complexity to your architecture.

Option D is incorrect because it requires you to create an AWS Lambda function and ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events. This solution has the following disadvantages:

? It does not guarantee that your Lambda function will always be warm, as Lambda may scale down your function if it does not receive any requests for a long period of time. This may cause cold start delays when your function is invoked by API Gateway.

? It incurs unnecessary costs, as you pay for the compute time of your Lambda function every time it is invoked by the EventBridge rule, even if it does not perform any useful work¹.

References:

? 1: AWS Lambda - Features

? 2: Amazon Elastic Container Service - Features

? 3: Amazon Elastic Kubernetes Service - Features

? [4]: Building API Gateway REST API with Lambda integration - Amazon API Gateway

? [5]: Improving latency with Provisioned Concurrency - AWS Lambda

? [6]: Integrating Amazon ECS with Amazon API Gateway - Amazon Elastic Container Service

? [7]: Integrating Amazon EKS with Amazon API Gateway - Amazon Elastic Kubernetes Service

? [8]: Managing concurrency for a Lambda function - AWS Lambda

NEW QUESTION 27

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

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

Which solution will meet these requirements MOST cost-effectively?

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

B. Configure a Hive metastore in Amazon EM

C. Migrate the existing on-premises Hive metastore into Amazon EM

D. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.

E. Configure an external Hive metastore in Amazon EM

F. Migrate the existing on-premises Hive metastore into Amazon EM

G. Use Amazon Aurora MySQL to store the company's data catalog.

H. Configure a new Hive metastore in Amazon EM

I. Migrate the existing on-premises Hive metastore into Amazon EM

J. Use the new metastore as the company's data catalog.

Answer: A

Explanation:

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

? Using AWS Database Migration Service

? Populating the AWS Glue Data Catalog

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.2: AWS Glue Data Catalog

NEW QUESTION 28

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 rewrites the 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 31

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 downtime 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 over many 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 32

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

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

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

A. AWS Glue

B. Amazon EMR

C. AWS Lambda

D. Amazon Redshift

Answer: A

Explanation:

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

? AWS Glue

? AWS Glue Data Catalog

? AWS Glue Workflows

? [AWS Glue Job Bookmarking]

? [AWS Glue Crawlers]

? [AWS Glue Schema Registry]

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

NEW QUESTION 33

A company has a production AWS account that runs company workloads. The company's security team created a security AWS account to store and analyze

security logs from the production AWS account. The security logs in the production AWS account are stored in Amazon CloudWatch Logs. The company needs to use Amazon Kinesis Data Streams to deliver the security logs to the security AWS account. Which solution will meet these requirements?

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

Answer: D

Explanation:

Amazon Kinesis Data Streams is a service that enables you to collect, process, and analyze real-time streaming data. You can use Kinesis Data Streams to ingest data from various sources, such as Amazon CloudWatch Logs, and deliver it to different destinations, such as Amazon S3 or Amazon Redshift. To use Kinesis Data Streams to deliver the security logs from the production AWS account to the security AWS account, you need to create a destination data stream in the security AWS account. This data stream will receive the log data from the CloudWatch Logs service in the production AWS account. To enable this cross-account data delivery, you need to create an IAM role and a trust policy in the security AWS account. The IAM role defines the permissions that the CloudWatch Logs service needs to put data into the destination data stream. The trust policy allows the production AWS account to assume the IAM role. Finally, you need to create a subscription filter in the production AWS account. A subscription filter defines the pattern to match log events and the destination to send the matching events. In this case, the destination is the destination data stream in the security AWS account. This solution meets the requirements of using Kinesis Data Streams to deliver the security logs to the security AWS account. The other options are either not possible or not optimal. You cannot create a destination data stream in the production AWS account, as this would not deliver the data to the security AWS account. You cannot create a subscription filter in the security AWS account, as this would not capture the log events from the production AWS account. References:

? Using Amazon Kinesis Data Streams with Amazon CloudWatch Logs

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.3: Amazon Kinesis Data Streams

NEW QUESTION 38

A data engineer must build an extract, transform, and load (ETL) pipeline to process and load data from 10 source systems into 10 tables that are in an Amazon Redshift database. All the source systems generate .csv, JSON, or Apache Parquet files every 15 minutes. The source systems all deliver files into one Amazon S3 bucket. The file sizes range from 10 MB to 20 GB. The ETL pipeline must function correctly despite changes to the data schema.

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

- A. Use an Amazon EventBridge rule to run an AWS Glue job every 15 minute
- B. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- C. Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minute
- D. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully
- E. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- F. Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket
- G. Configure an AWS Glue job to process and load the data into the Amazon Redshift table
- H. Create a second Lambda function to run the AWS Glue job
- I. Create an Amazon EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.
- J. Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket
- K. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully
- L. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- M. Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket
- N. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFrame
- O. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream
- P. Configure the delivery stream to load data into the Amazon Redshift tables.

Answer: AB

Explanation:

Using an Amazon EventBridge rule to run an AWS Glue job or invoke an AWS Glue workflow job every 15 minutes are two possible solutions that will meet the requirements. AWS Glue is a serverless ETL service that can process and load data from various sources to various targets, including Amazon Redshift. AWS Glue can handle different data formats, such as CSV, JSON, and Parquet, and also support schema evolution, meaning it can adapt to changes in the data schema over time. AWS Glue can also leverage Apache Spark to perform distributed processing and transformation of large datasets. AWS Glue integrates with Amazon EventBridge, which is a serverless event bus service that can trigger actions based on rules and schedules. By using an Amazon EventBridge rule, you can invoke an AWS Glue job or workflow every 15 minutes, and configure the job or workflow to run an AWS Glue crawler and then load the data into the Amazon Redshift tables. This way, you can build a cost-effective and scalable ETL pipeline that can handle data from 10 source systems and function correctly despite changes to the data schema.

The other options are not solutions that will meet the requirements. Option C, configuring an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket, and creating a second Lambda function to run the AWS Glue job, is not a feasible solution, as it would require a lot of Lambda invocations and coordination. AWS Lambda has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the ETL pipeline. Option D, configuring an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket, is not a necessary solution, as you can use an Amazon EventBridge rule to invoke the AWS Glue workflow directly, without the need for a Lambda function. Option E, configuring an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket, and configuring the AWS Glue job to put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream, is not a cost-effective solution, as it would incur additional costs for Lambda invocations and data delivery. Moreover, using Amazon Kinesis Data Firehose to load data into Amazon Redshift is not suitable for frequent and small batches of data, as it can cause performance issues and data fragmentation. References:

? AWS Glue

? Amazon EventBridge

? Using AWS Glue to run ETL jobs against non-native JDBC data sources

? [AWS Lambda quotas]

? [Amazon Kinesis Data Firehose quotas]

NEW QUESTION 40

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

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

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

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

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

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 44

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

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog. Which solution will meet these requirements?

- A. Create an IAM role that includes the AmazonS3FullAccess policy
- B. Associate the role with the crawler
- C. Specify the S3 bucket path of the source data as the crawler's data store
- D. Create a daily schedule to run the crawler
- E. Configure the output destination to a new path in the existing S3 bucket.
- F. Create an IAM role that includes the AWSGlueServiceRole policy
- G. Associate the role with the crawler
- H. Specify the S3 bucket path of the source data as the crawler's data store
- I. Create a daily schedule to run the crawler
- J. Specify a database name for the output.
- K. Create an IAM role that includes the AmazonS3FullAccess policy
- L. Associate the role with the crawler
- M. Specify the S3 bucket path of the source data as the crawler's data store
- N. Allocate data processing units (DPU) to run the crawler every day
- O. Specify a database name for the output.
- P. Create an IAM role that includes the AWSGlueServiceRole policy
- Q. Associate the role with the crawler
- R. Specify the S3 bucket path of the source data as the crawler's data store
- S. Allocate data processing units (DPU) to run the crawler every day
- T. Configure the output destination to a new path in the existing S3 bucket.

Answer: B

Explanation:

To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:

? Create an IAM role that has the necessary permissions to access the S3 data and

the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions¹.

? Associate the role with the crawler.

? Specify the S3 bucket path of the source data as the crawler's data store. The crawler will scan the data and infer the schema and format².

? Create a daily schedule to run the crawler. The crawler will run at the specified time every day and update the Data Catalog with any changes in the data³.

? Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in the S3 bucket, such as the location, schema, and classification.

Option B is the only solution that includes all these steps. Therefore, option B is the correct answer.

Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data Catalog.

Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.

Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.

References:

? 1: AWS managed (predefined) policies for AWS Glue - AWS Glue

? 2: Data Catalog and crawlers in AWS Glue - AWS Glue

? 3: Scheduling an AWS Glue crawler - AWS Glue

? [4]: Parameters set on Data Catalog tables by crawler - AWS Glue

? [5]: AWS Glue pricing - Amazon Web Services (AWS)

NEW QUESTION 47

A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during non-business hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to comply with company policies.

A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.

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

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

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 Amazon Athena 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 50

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

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

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

- A. Turn on the public access setting for the DB instance.
- B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.
- C. Configure the Lambda function to run in the same subnet that the DB instance uses.
- D. Attach the same security group to the Lambda function and the DB instance.
- 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 55

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 suitable for 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 58

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 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, 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 59

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

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

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

Answer: B

Explanation:

AWS Lake Formation is a service that allows you to easily set up, secure, and manage data lakes. One of the features of Lake Formation is row-level security, which enables you to control access to specific rows or columns of data based on the identity or role of the user. This feature is useful for scenarios where you need to restrict access to sensitive or regulated data, such as customer data from different countries. By registering the S3 bucket as a data lake location in Lake Formation, you can use the Lake Formation console or APIs to define and apply row-level security policies to the data in the bucket. You can also use Lake Formation blueprints to automate the ingestion and transformation of data from various sources into the data lake. This solution requires the least operational effort compared to the other options, as it does not involve creating or moving data, or managing multiple tables, views, or roles. References:

? AWS Lake Formation

? Row-Level Security

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.2: AWS Lake Formation

NEW QUESTION 61

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

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

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

- A. Create an AWS Glue partition index
- B. Enable partition filtering.
- C. Bucket the data based on a column that the 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 entire table 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 64

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

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

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

Answer: B

Explanation:

Option B is the best solution to meet the requirements with the least operational overhead because S3 Select is a feature that allows you to retrieve only a subset of data from an S3 object by using simple SQL expressions. S3 Select works on objects stored in CSV, JSON, or Parquet format. By using S3 Select, you can avoid the need to download and process the entire S3 object, which reduces the amount of data transferred and the computation time. S3 Select is also easy to use and does not require any additional services or resources.

Option A is not a good solution because it involves writing custom code and configuring an AWS Lambda function to load data from the S3 bucket into a pandas dataframe and query the required column. This option adds complexity and latency to the data retrieval process and requires additional resources and configuration. Moreover, AWS Lambda has limitations on the execution time, memory, and concurrency, which may affect the performance and reliability of the data retrieval process.

Option C is not a good solution because it involves creating and running an AWS Glue DataBrew project to consume the S3 objects and query the required column. AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. However, in this scenario, the data is already in Parquet format, which is a columnar storage format that is optimized for analytics. Therefore, there is no need to use AWS Glue DataBrew to prepare the data. Moreover, AWS Glue DataBrew adds extra time and cost to the data retrieval process and requires additional resources and configuration.

Option D is not a good solution because it involves running an AWS Glue crawler on the S3 objects and using a SQL SELECT statement in Amazon Athena to query the required column. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. However, in this scenario, the schema and format of the data are already known and fixed, so there is no need to run a crawler to discover them. Moreover, running a crawler and using Amazon Athena adds extra time and cost to the data retrieval process and requires additional services and configuration.

References:

- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? S3 Select and Glacier Select - Amazon Simple Storage Service
- ? AWS Lambda - FAQs
- ? What Is AWS Glue DataBrew? - AWS Glue DataBrew
- ? Populating the AWS Glue Data Catalog - AWS Glue
- ? What is Amazon Athena? - Amazon Athena

NEW QUESTION 65

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

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

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

Answer: A

Explanation:

AWS DataSync is an online data movement and discovery service that simplifies and accelerates data migrations to AWS as well as moving data to and from on-premises storage, edge locations, other cloud providers, and AWS Storage services¹. AWS DataSync can copy data to and from various sources and targets, including Amazon S3, and handle files in multiple formats. AWS DataSync also supports incremental transfers, meaning it can detect and copy only the changes to the data, reducing the amount of data transferred and improving the performance. AWS DataSync can automate and schedule the transfer process using triggers, and monitor the progress and status of the transfers using CloudWatch metrics and events¹.

AWS DataSync is the most operationally efficient way to transfer the data in this scenario, as it meets all the requirements and offers a serverless and scalable

solution. AWS Glue, AWS Direct Connect, and Amazon S3 Transfer Acceleration are not the best options for this scenario, as they have some limitations or drawbacks compared to AWS DataSync. AWS Glue is a serverless ETL service that can extract, transform, and load data from various sources to various targets, including Amazon S3. However, AWS Glue is not designed for large-scale data transfers, as it has some quotas and limits on the number and size of files it can process³. AWS Glue also does not support incremental transfers, meaning it would have to copy the entire data set every time, which would be inefficient and costly.

AWS Direct Connect is a service that establishes a dedicated network connection between your on-premises data center and AWS, bypassing the public internet and improving the bandwidth and performance of the data transfer. However, AWS Direct Connect is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS DataSync, AWS Storage Gateway, or AWS CLI. AWS Direct Connect also has some hardware and location requirements, and charges you for the port hours and data transfer out of AWS.

Amazon S3 Transfer Acceleration is a feature that enables faster data transfers to Amazon S3 over long distances, using the AWS edge locations and optimized network paths. However, Amazon S3 Transfer Acceleration is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS CLI, AWS SDK, or third-party software. Amazon S3 Transfer Acceleration also charges you for the data transferred over the accelerated endpoints, and does not guarantee a performance improvement for every transfer, as it depends on various factors such as the network conditions, the distance, and the object size. References:

? AWS DataSync

? AWS Glue

? AWS Glue quotas and limits

? [AWS Direct Connect]

? [Data transfer options for AWS Direct Connect]

? [Amazon S3 Transfer Acceleration]

? [Using Amazon S3 Transfer Acceleration]

NEW QUESTION 66

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

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

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

Answer: BD

Explanation:

The best combination of resources to meet the requirements of high reliability, cost-optimization, and performance for running Apache Spark jobs on Amazon EMR is to use Amazon S3 as a persistent data store and Graviton instances for core nodes and task nodes.

Amazon S3 is a highly durable, scalable, and secure object storage service that can store any amount of data for a variety of use cases, including big data analytics¹. Amazon S3 is a better choice than HDFS as a persistent data store for Amazon EMR, as it decouples the storage from the compute layer, allowing for more flexibility and cost-efficiency. Amazon S3 also supports data encryption, versioning, lifecycle management, and cross-region replication¹. Amazon EMR integrates seamlessly with Amazon S3, using EMR File System (EMRFS) to access data stored in Amazon S3 buckets². EMRFS also supports consistent view, which enables Amazon EMR to provide read-after-write consistency for Amazon S3 objects that are accessed through EMRFS².

Graviton instances are powered by Arm-based AWS Graviton² processors that deliver up to 40% better price performance over comparable current generation x86-based instances³. Graviton instances are ideal for running workloads that are CPU-bound, memory-bound, or network-bound, such as big data analytics, web servers, and open- source databases³. Graviton instances are compatible with Amazon EMR, and can be used for both core nodes and task nodes. Core nodes are responsible for running the data processing frameworks, such as Apache Spark, and storing data in HDFS or the local file system. Task nodes are optional nodes that can be added to a cluster to increase the processing power and throughput. By using Graviton instances for both core nodes and task nodes, you can achieve higher performance and lower cost than using x86-based instances.

Using Spot Instances for all primary nodes is not a good option, as it can compromise the reliability and availability of the cluster. Spot Instances are spare EC2 instances that are available at up to 90% discount compared to On-Demand prices, but they can be interrupted by EC2 with a two-minute notice when EC2 needs the capacity back. Primary nodes are the nodes that run the cluster software, such as Hadoop, Spark, Hive, and Hue, and are essential for the cluster operation. If a primary node is interrupted by EC2, the cluster will fail or become unstable. Therefore, it is recommended to use On-Demand Instances or Reserved Instances for primary nodes, and use Spot Instances only for task nodes that can tolerate interruptions. References:

? Amazon S3 - Cloud Object Storage

? EMR File System (EMRFS)

? AWS Graviton2 Processor-Powered Amazon EC2 Instances

? [Plan and Configure EC2 Instances]

? [Amazon EC2 Spot Instances]

? [Best Practices for Amazon EMR]

NEW QUESTION 71

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

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

- A. Use Amazon EMR and Apache Ranger.
- B. Use a Hive metastore on an EMR cluster.
- C. Use the AWS Glue Data Catalog.
- D. Use a metastore on an Amazon RDS for MySQL DB instance.

Answer: C

Explanation:

The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS

instance, or migrating the metadata manually. References:

- ? Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying AWS Glue Data Catalog as the metastore)
- ? Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)
- ? AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

NEW QUESTION 76

A company receives call logs as Amazon S3 objects that contain sensitive customer information. The company must protect the S3 objects by using encryption. The company must also use encryption keys that only specific employees can access. Which solution will meet these requirements with the LEAST effort?

- A. Use an AWS CloudHSM cluster to store the encryption key
- B. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the object
- C. Deploy an IAM policy that restricts access to the CloudHSM cluster.
- D. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer informatio
- E. Restrict access to the keys that encrypt the objects.
- F. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer informatio
- G. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.
- H. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer informatio
- I. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Answer: C

Explanation:

Option C is the best solution to meet the requirements with the least effort because server-side encryption with AWS KMS keys (SSE-KMS) is a feature that allows you to encrypt data at rest in Amazon S3 using keys managed by AWS Key Management Service (AWS KMS). AWS KMS is a fully managed service that enables you to create and manage encryption keys for your AWS services and applications. AWS KMS also allows you to define granular access policies for your keys, such as who can use them to encrypt and decrypt data, and under what conditions. By using SSE-KMS, you can protect your S3 objects by using encryption keys that only specific employees can access, without having to manage the encryption and decryption process yourself.

Option A is not a good solution because it involves using AWS CloudHSM, which is a service that provides hardware security modules (HSMs) in the AWS Cloud. AWS CloudHSM allows you to generate and use your own encryption keys on dedicated hardware that is compliant with various standards and regulations. However, AWS CloudHSM is not a fully managed service and requires more effort to set up and maintain than AWS KMS. Moreover, AWS CloudHSM does not integrate with Amazon S3, so you have to configure the process that writes to S3 to make calls to CloudHSM to encrypt and decrypt the objects, which adds complexity and latency to the data protection process. Option B is not a good solution because it involves using server-side encryption with customer-provided keys (SSE-C), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that you provide and manage yourself. SSE-C requires you to send your encryption key along with each request to upload or retrieve an object. However, SSE-C does not provide any mechanism to restrict access to the keys that encrypt the objects, so you have to implement your own key management and access control system, which adds more effort and risk to the data protection process.

Option D is not a good solution because it involves using server-side encryption with Amazon S3 managed keys (SSE-S3), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that are managed by Amazon S3. SSE-S3 automatically encrypts and decrypts your objects as they are uploaded and downloaded from S3. However, SSE-S3 does not allow you to control who can access the encryption keys or under what conditions. SSE-S3 uses a single encryption key for each S3 bucket, which is shared by all users who have access to the bucket. This means that you cannot restrict access to the keys that encrypt the objects by specific employees, which does not meet the requirements.

References:

- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? Protecting Data Using Server-Side Encryption with AWS KMS–Managed Encryption Keys (SSE-KMS) - Amazon Simple Storage Service
- ? What is AWS Key Management Service? - AWS Key Management Service
- ? What is AWS CloudHSM? - AWS CloudHSM
- ? Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C) - Amazon Simple Storage Service
- ? Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) - Amazon Simple Storage Service

NEW QUESTION 79

A company stores data in a data lake that is in Amazon S3. Some data that the company stores in the data lake contains personally identifiable information (PII). Multiple user groups need to access the raw data. The company must ensure that user groups can access only the PII that they require. Which solution will meet these requirements with the LEAST effort?

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

Answer: A

Explanation:

Amazon Athena is a serverless, interactive query service that enables you to analyze data in Amazon S3 using standard SQL. AWS Lake Formation is a service that helps you build, secure, and manage data lakes on AWS. You can use AWS Lake Formation to create data filters that define the level of access for different IAM roles based on the columns, rows, or tags of the data. By using Amazon Athena to query the data and AWS Lake Formation to create data filters, the company can meet the requirements of ensuring that user groups can access only the PII that they require with the least effort. The solution is to use Amazon Athena to query the data in the data lake that is in Amazon S3. Then, set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. For example, a data filter can allow a user group to access only the columns that contain the PII that they need, such as name and email address, and deny access to the columns that contain the PII that they do not need, such as phone number and social security number. Finally, assign each user to the IAM role that matches the user's PII access requirements. This way, the user groups can access the data in the data lake securely and efficiently. The other

options are either not feasible or not optimal. Using Amazon QuickSight to access the data (option B) would require the company to pay for the QuickSight service and to configure the column-level security features for each user. Building a custom query builder UI that will run Athena queries in the background to access the data (option C) would require the company to develop and maintain the UI and to integrate it with Amazon Cognito. Creating IAM roles that have different levels of granular access (option D) would require the company to manage multiple IAM roles and policies and to ensure that they are aligned with the data schema.

References:

? Amazon Athena

? AWS Lake Formation

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

NEW QUESTION 80

.....

Thank You for Trying Our Product

* **100% Pass or Money Back**

All our products come with a 90-day Money Back Guarantee.

* **One year free update**

You can enjoy free update one year. 24x7 online support.

* **Trusted by Millions**

We currently serve more than 30,000,000 customers.

* **Shop Securely**

All transactions are protected by VeriSign!

100% Pass Your AWS-Certified-Data-Engineer-Associate Exam with Our Prep Materials Via below:

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